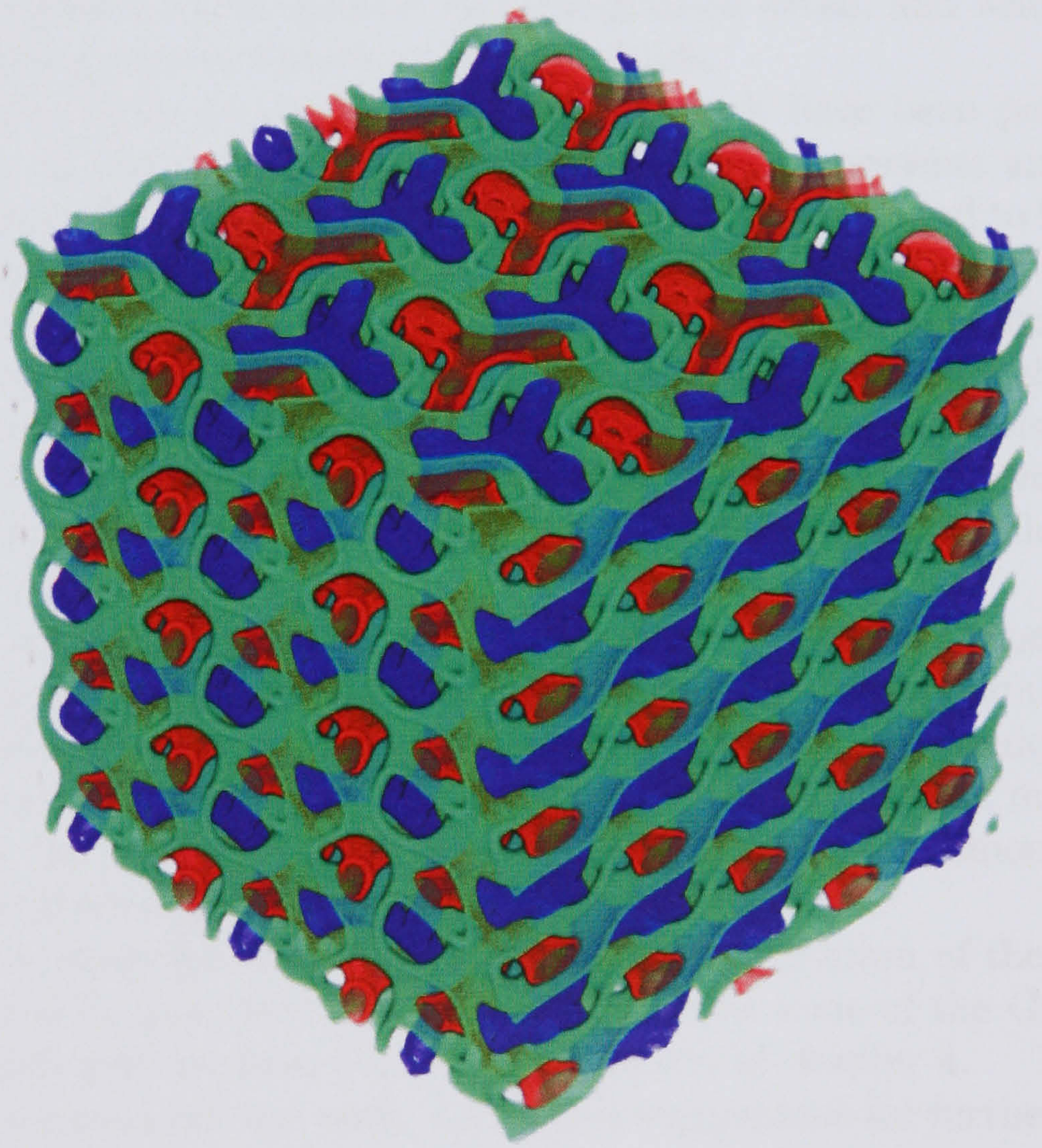
Mesoscale fluid simulation with the Lattice Boltzmann method

Chin, Jonathan

Mesoscale fluid simulation with the Lattice Boltzmann method

Jonathan Chin <jon-thesis@earth.li>



A Thesis submitted for the degree of Doctor of Philosophy
Queen Mary, University of London
2005

Abstract

This thesis describes investigations of several complex fluid effects, including hydrodynamic spinodal decomposition, viscous instability, and self-assembly of a cubic surfactant phase, by simulating them with a lattice Boltzmann computational model.

The introduction describes what is meant by the term "complex fluid", and why such fluids are both important and difficult to understand. A key feature of complex fluids is that their behaviour spans length and time scales. The lattice Boltzmann method is presented as a modelling technique which sits at a "mesoscale" level intermediate between coarse-grained and fine-grained detail, and which is therefore ideal for modelling certain classes of complex fluids.

The following chapters describe simulations which have been performed using this technique, in two and three dimensions. Chapter 2 presents an investigation into the separation of a mixture of two fluids. This process is found to involve several physical mechanisms at different stages. The simulated behaviour is found to be in good agreement with existing theory, and a curious effect, due to multiple competing mechanisms, is observed, in agreement with experiments and other simulations.

Chapter 3 describes an improvement to lattice Boltzmann models of Hele-Shaw flow, along with simulations which quantitatively demonstrate improvements in both accuracy and numerical stability. The Saffman-Taylor hydrodynamic instability is demonstrated using this model.

Chapter 4 contains the details and results of the TeraGyroid experiment, which involved extremely large-scale simulations to investigate the dynamical behaviour of a self-assembling structure. The first finite-size-effect-free dynamical simulations of such a system are presented. It is found that several different mechanisms are responsible for the assembly; the existence of chiral domains is demonstrated, along with an examination of domain growth during self-assembly.

Appendix A describes some aspects of the implementation of the lattice Boltzmann codes used in this thesis; appendix B describes some of the Grid computing techniques which were necessary for the simulations of chapter 4.

Chapter 5 summarises the work, and makes suggestions for further research and improvement.

Contents

List of Tables

Acknowledgements

I would like to dedicate this thesis to my parents, for all their support and encouragement during its creation.

I am extremely grateful to my supervisor, Peter Coveney, for his guidance during this work and in particular for his extreme patience while it was written up. In addition, I have benefited from many stimulating and enjoyable collaborations and discussions with Edo Boek, Jean-Pierre Boon, Bruce Boghosian, Simon Clifford, Nelido González , Jens Harting, Matt Harvey, John Kendrick, Thomas Lewiner, Peter Love, Mark McKeown, Maziar Nekovee, Stephen Pickles, Kevin Roy, and Maddalena Venturoli.

Thanks must go to Huntsman Corporation, Queen Mary, University of London, and Schlumberger Cambridge Research, for generous financial support.

I am indebted to far too many people from `earth.li`, from the nebulously defined but relentlessly fun London Kitten Herd, and from the denizens of That Channel, for encouragement and for making life enjoyable during my PhD; specific thanks are due to, amongst others, Chris Ball, Tim Bond, Arthur Bullard, Matthew Byng-Maddick, Nicholas Clark, Simon Cozens, Ben Evans, Matthew Garrett, Marna Gilligan, Pete Gillin, Alex Gough, Juliet Kemp, Martin Ling, Andrew Loveridge, Eoin Malins, Jonathan McDowell, Sean Purdy, and Ali Young.

Finally, I am grateful to Kake, for love, support, and wisdom.

This thesis is the product of my own work, unless otherwise stated. It is based in part on work described in the following refereed publications.

- J. Chin, J. Harting, S. Jha, P. V. Coveney, A. R. Porter, and S. M. Pickles. 'Steering in computational science: mesoscale modelling and simulation'. *Contemporary Physics*, 44(5) 417–434 (2003). doi:10.1080/0010751031000160504 6

- J. Chin and P. V. Coveney. 'Lattice Boltzmann study of spinodal decomposition in two dimensions'. *Phys. Rev. E*, 66(016303) (2002). doi:10.1103/PhysRevE.66.016303

- J. Chin, E. S. Boek, and P. V. Coveney. 'Lattice Boltzmann simulation of the flow of binary immiscible fluids with different viscosities using the Shan-Chen microscopic interaction model'. *Phil. Trans.: Mat. Phys. Eng. Sci.*, 360(1792) 547–558 (2002). doi:10.1098/rsta.2001.0953

- J. Harting, M. J. Harvey, J. Chin, and P. V. Coveney. 'Detection and tracking of defects in the gyroid mesophase'. *Comp. Phys. Comm.*, 165(2) 97–109 (2005). doi:10.1016/j.cpc.2004.10.001

- P. Grosfils, J. P. Boon, J. Chin, and E. S. Boek. 'Structural and dynamical characterization of Hele-Shaw viscous fingering'. *Phil. Trans.: Mat. Phys. Eng. Sci.*, 362(1821) 1471–2962 (2004). doi:10.1098/rsta.2004.1398

- P. J. Love, M. Nekovee, P. V. Coveney, J. Chin, N. González-Segredo, and J. M. R. Martin. 'Simulations of amphiphilic fluids using mesoscale lattice-Boltzmann and lattice-gas methods'. *Comp. Phys. Comm.*, 153 340–358 (2003). doi:10.1016/S0010-4655(03)00200-5

- E. S. Boek, J. Chin, and P. V. Coveney. 'Lattice Boltzmann simulation of the flow of non-Newtonian fluids in porous media'. *Int. J. Mod. Phys. B*, 17(1–2) 99–102 (2003). doi:10.1142/S021797920301714X

- M. Nekovee, J. Chin, and P. V. Coveney. 'Massively parallel mesoscopic simulations of complex fluids'. In 'Proceedings of the Simulation 99 UCLse/UK Conference on Simulation', (1999)

- M. Nekovee, J. Chin, N. González-Segredo, and P. V. Coveney. 'A parallel lattice-Boltzmann method for large scale simulations of complex fluids'. In 'Proceedings of the Fourth UNAM Supercomputing Conference, Singapore'. (1999). URL `http://www.chem.ucl.ac.uk/ccs/unam.ps.gz`

"There is no exercise of the intellect which is not, in the final analysis, useless. A philosophical doctrine begins as a plausible description of the universe: with the passage of the years it becomes a mere chapter — if not a paragraph or a name — in the history of philosophy."

– Jorge Luis Borges

Chapter 1

Introduction

This chapter introduces the concept of mesoscale modelling, particularly in the context of fluid dynamics. A brief overview is given of how the macroscopic Navier-Stokes equations may be obtained from the microscopic equations governing fluid molecules, by first obtaining an intermediate level of description, called the Boltzmann equation. Some computational equivalents of these equations are presented, including the lattice Boltzmann equation (LBE), the discrete equivalent of the Boltzmann equation. Some variants of these models, which permit simulation of interacting fluid mixtures, are then discussed, with a focus on the Shan-Chen LBE model which is used in the rest of this thesis.

1.1 Mesoscale modelling

Physical science has met with much success in modelling phenomena on microscopic and macroscopic scales. A microscopic model usually considers the behaviour of every single component atom of a system; the increasing complexity of such models as the number of atoms increases tends to limit the treatable length scales to the order of nanometres; typical timescales are limited to the order of nanoseconds. For example, a microscopic view of a paramagnetic crystal would describe the magnetization of every single atom in the crystal lattice; a microscopic model of a fluid would describe the position, momentum, and rotation of every single fluid molecule, which would comprise of order 10^{17} variables for a single cubic millimetre of monatomic gas at room temperature and pressure.

A macroscopic model, on the other hand, usually deals with a smaller number of parameters, often more closely related to physical observables. A macroscopic view of a paramagnetic crystal would describe the nett magnetization of the entire crystal and its temperature; a macroscopic view of a fluid would describe its velocity, density and temperature at various points.

The field of Statistical Mechanics has met with much success in producing macroscopic, thermodynamic descriptions of systems from their microscopic descriptions; however, its success has been limited in cases where the system is not close to thermodynamic equilibrium. In addition, comparatively few methods exist for the treatment of systems which contain processes operating at a wide range of length scales.

Mesoscale models are intended to describe systems which have intermediate length or time scales, or which contain processes operating at several distinct scales. "Complex fluids", such as fluid mixtures, colloids, suspensions, or fluids flowing

through porous media, are good examples of an area where mesoscale modelling is important, since the bulk fluid flow is an important macroscopic property, but its behaviour is highly dependent on the microscopic behaviour of the fluid interface or suspended particles. That a mixture of cornflour and water becomes more difficult to stir when one stirs it quickly is macroscopic behaviour often observed in the kitchen, but it is due to the microscopic interaction between the tiny starch molecules in the cornflour.

Bulk fluid flow, or hydrodynamics, is one of the hardest macroscopic effects to deal with: the equations of hydrodynamic flow are nonlinear, and even dedicated macroscopic treatments run into problems when the flow starts to become turbulent. Because hydrodynamic effects involve very large numbers of atoms flowing in an organized manner (rather than, say, the random-walk behaviour seen in diffusion), they are inaccessible through microscopic techniques such as molecular dynamics, which are far too expensive. However, as will be outlined in the following section, it is possible to show that despite the differences in scale, hydrodynamic effects are nonetheless a direct consequence of the vast number of interparticle interactions at the microscopic scale.

1.2 Fluid dynamics from the ground up

The most detailed view of a fluid permitted while remaining within the realm of classical physics is that of a collection of many tiny particles, each with a well-defined position $\mathbf{x}$ and momentum $\mathbf{p}$, amongst other internal degrees of freedom; these particles may move under the influence of an externally imposed force field, such as gravity, and interact with one another, such as through electromagnetic forces. For a single-component fluid, this view is often reduced to one of point particles with position and momentum, moving under the influence of an external force, and a pairwise interaction potential $\Phi_{ij} = \Phi(|\mathbf{x}_i - \mathbf{x}_j|)$. A system of N such particles will have a total kinetic energy T and total potential energy V; their motion is then governed by the Hamiltonian

$$H = T + V = \sum_{i=1}^{N} \frac{p_i^2}{2m} + \sum_{i=1}^{N} \sum_{j>i} \Phi_{ij} \tag{1.1}$$

through Hamilton's equations,

$$\dot{\mathbf{p}}_i = -\frac{\partial H}{\partial \mathbf{x}_i} \quad ; \quad \dot{\mathbf{x}}_i = \frac{\partial H}{\partial \mathbf{p}_i}.$$

This representation contains $6N$ equations in $6N$ unknowns. If the transformation $t' = -t; \mathbf{p}'_i = -\mathbf{p}_i; \mathbf{x}'_i = \mathbf{x}_i$ is made, corresponding to reversing time and velocities, then the transformed system will also obey Hamilton's equations, showing that the system is time-reversible.

At the other extreme, the least detailed description which is still readily recognisable as that of a fluid contains the continuum velocity field $\mathbf{v}(\mathbf{x})$ and density $\rho(\mathbf{x})$ or pressure $p(\mathbf{x})$. For a typical Newtonian fluid, these will be related by the Navier-Stokes equations, which, for an incompressible fluid, take the form

$$\dot{\mathbf{v}} + (\mathbf{v} \cdot \nabla)\,\mathbf{v} + \frac{1}{\rho}\nabla p = \nu \nabla^2 \mathbf{v} \quad : \quad \nabla \cdot \mathbf{v} = 0 \tag{1.2}$$

where the kinematic viscosity is ν; this representation contains the velocity and pressure fields as functions of position $\mathbf{x}$. The Navier-Stokes equations are closed in the macroscopic variables: while they describe the flow of a fluid, they say nothing and know nothing about the composition of the fluid or the origin of effects such as viscosity. [1]

One of the triumphs of modern kinetic theory has been the demonstration that it is possible to move from the microscopic Hamiltonian description to the macroscopic Navier-Stokes description, by carefully integrating out irrelevant degrees of freedom.

This procedure begins by describing the evolution of the system in $6N$-dimensional phase space, using the Liouville equation. This is then recast as a chain of successively more high-level distribution functions, which is truncated to produce the Boltzmann equation. The Boltzmann equation is then simplified to the BGK equation while retaining most of its important features; it can then be shown that the BGK equation produces Navier-Stokes behaviour on a coarse-grained scale.

1.2.1 The Liouville Equation

An N-particle system will, at any single point in time, occupy a single point in $6N$-dimensional phase space. Without sacrificing any information, let the system be described in terms of the N-particle distribution function $\psi(\mathbf{x}_1, \mathbf{p}_1, \ldots, \mathbf{x}_N, \mathbf{p}_N; t)$, normalized to unity, which can be regarded as the probability of finding particle 1 with position $\mathbf{x}_1$ and momentum $\mathbf{p}_1$, particle 2 with position $\mathbf{x}_2$ and momentum $\mathbf{p}_2$, etc. For brevity, the combined position and momentum of a particle may be written $\mathbf{z}_i = (\mathbf{x}_i, \mathbf{p}_i)$. For a single system whose state is known exactly, ψ will be a delta-function spike at the exact values of the coordinates $\mathbf{z}_i$, evolving according to the Hamiltonian (1.1). A continuous, real-valued ψ can represent the evolution of a microcanonical ensemble of such systems. Since probability (or the number of systems in the ensemble) is conserved,

$$\frac{\partial \psi}{\partial t} + \sum_i \frac{\partial}{\partial \mathbf{z}_i} \cdot (\psi \dot{\mathbf{z}}_i) = 0. \tag{1.3}$$

Substitution of Hamilton's equations simplifies this to

$$\left(\frac{\partial}{\partial t} + \sum_i \dot{\mathbf{z}}_i \cdot \frac{\partial}{\partial \mathbf{z}_i} \right) \psi = 0, \tag{1.4}$$

a result normally referred to as Liouville's theorem, showing that the phase space probability density function ψ behaves like an incompressible fluid. It is important to note that this representation of the dynamics remains exact (no information has been lost) and reversible.

1.2.2 The BBGKY Hierarchy

The Liouville equation contains some $6N$ degrees of freedom; in order to obtain a lower-dimensional and more tractable (if less precise) description of the system, some of these degrees of freedom must be integrated or averaged out.

The single particle distribution function $f_1(\mathbf{x}, \mathbf{p}, t)$ describes the density of particles at position $\mathbf{x}$ with momentum $\mathbf{p}$ at time t, and is normalized in such a way that $\int f_1 \, \mathrm{d}\mathbf{z} = N$, in contrast to ψ, which, being a probability, is normalized to one. f_1 can be found by integrating out all the other degrees of freedom:

$$f_1(\mathbf{x}'_1, \mathbf{p}'_1, t) = f_1(\mathbf{z}'_1, t) \doteq N \int \psi(\mathbf{z}'_1, \mathbf{z}_2, \ldots \mathbf{z}_N)\, \mathrm{d}\mathbf{z}_2 \cdots \mathrm{d}\mathbf{z}_N \tag{1.5}$$

A description of the system in terms of f_1 would neglect correlations between pairs of particles, which are described by the two-particle distribution function $f_2(\mathbf{z}_1, \mathbf{z}_2)$, which in turn neglects three-particle correlations given by f_3, and so on. In general, the n-particle distribution function is defined as

$$f_n(\mathbf{z}'_1, \ldots, \mathbf{z}'_n, t) \doteq \frac{N!}{(N-n)!} \int \psi(\mathbf{z}'_1, \ldots \mathbf{z}'_n, \mathbf{z}_{n+1}, \ldots \mathbf{z}_N)\, \mathrm{d}\mathbf{z}_{n+1} \cdots \mathrm{d}\mathbf{z}_N. \tag{1.6}$$

The BBGKY hierarchy is a cascade of equations derived from the Liouville equation, each describing the dynamics of the n-particle distribution function as an integral over the $(n+1)$-particle distribution function. It was developed independently by Bogoliubov, Born, Green, Kirkwood, and Yvon[2, 3, 4, 5, 6]; detailed examinations of the hierarchy can be found in, for example, Liboff[7] or Moore[8].

The second term in the operator acting on ψ in the Liouville equation (1.4) is often called the Liouvillian operator $\mathcal{L}$, and can be written out in full as

$$\mathcal{L}_N(\mathbf{z}_1, \ldots, \mathbf{z}_N) \doteq \sum_{i=1}^{N} \frac{1}{m} \mathbf{p}_i \cdot \frac{\partial}{\partial \mathbf{x}_i} + \sum_{i=1}^{N} \sum_{j>i} \mathbf{K}_{ij} \cdot \left(\frac{\partial}{\partial \mathbf{p}_i} - \frac{\partial}{\partial \mathbf{p}_j} \right), \tag{1.7}$$

where the force $\mathbf{K}_{ij}$ exerted on particle i by particle j is

$$\mathbf{K}_{ij} \doteq -\frac{\partial}{\partial \mathbf{x}_i} \Phi\left(|\mathbf{x}_i - \mathbf{x}_j|\right). \tag{1.8}$$

The Liouvillian can be split into three terms: the first representing particles 1 to n and their interactions with one another, the second representing particles $(n+1)$ to N and their interactions, and the third representing the interactions between particles in the first set and particles in the second.

$$\mathcal{L}_N(\mathbf{z}_1, \ldots, \mathbf{z}_N) = \mathcal{L}_n(\mathbf{z}_1, \ldots, \mathbf{z}_n) + \mathcal{L}_{N-n}(\mathbf{z}_{n+1}, \ldots, \mathbf{z}_N) + \sum_{i=1}^{n} \sum_{j=n+1}^{N} \mathbf{K}_{ij} \cdot \left(\frac{\partial}{\partial \mathbf{p}_i} - \frac{\partial}{\partial \mathbf{p}_j} \right) \tag{1.9}$$

Taking the Liouville equation $(\partial_t + \mathcal{L})\psi = 0$ and integrating over the coordinates $\mathbf{z}_{n+1} \ldots \mathbf{z}_N$ allows the middle term in the expanded form (1.9) of the Liouvillian to be written entirely in terms of surface integrals, so that it drops out to leave

$$\int \left(\frac{\partial}{\partial t} + \mathcal{L}_n \right) \psi\, \mathrm{d}\mathbf{z}_{n+1} \ldots \mathrm{d}\mathbf{z}_N = -\int \sum_{i=1}^{n} \sum_{j=n+1}^{N} \mathbf{K}_{ij} \left(\frac{\partial}{\partial \mathbf{p}_i} - \frac{\partial}{\partial \mathbf{p}_j} \right) \psi\, \mathrm{d}\mathbf{z}_{n+1} \ldots \mathrm{d}\mathbf{z}_N \tag{1.10}$$

Some further algebra to simplify and recast this in terms of distribution functions yields the famous chain of equations

$$\left(\frac{\partial}{\partial t} + \mathcal{L}_n \right) f_n = -\sum_{i=1}^{n} \int \mathbf{K}_{i,n+1} \cdot \frac{\partial}{\partial \mathbf{p}_i} f_{n+1}\, \mathrm{d}\mathbf{z}_{n+1}. \tag{1.11}$$

The complete set of BBGKY equations is still exact (and time-reversible): however, one cannot simply pluck out the lowest-order equation for f_1 and solve it. since this requires knowledge of f_2 from the next equation up. Some approximation or external information is required in order to close the equations and produce a standalone description which can be solved. There are many ways of doing so. which all end up at the Boltzmann equation: the following procedure uses a simple virial expansion, but a simpler approach (which is a little less clear about the necessary assumptions) was used by Born and Green[2], and a much more detailed approach is given in Liboff[7].

1.2.3 The Boltzmann Equation

The two highest-level BBGKY equations, with $n = 1$ and $n = 2$, are:

$$\left(\frac{\partial}{\partial t} + \frac{1}{m}\mathbf{p}_1 \cdot \frac{\partial}{\partial \mathbf{x}_1}\right) f_1 = -\int \mathbf{K}_{12} \cdot \frac{\partial}{\partial \mathbf{p}_1} f_2 \, \mathrm{d}\mathbf{z}_2 \tag{1.12}$$

and

$$\left[\frac{\partial}{\partial t} + \sum_{i=1}^{2} \frac{1}{m}\mathbf{p}_i \cdot \frac{\partial}{\partial \mathbf{x}_i} + \mathbf{K}_{12} \cdot \left(\frac{\partial}{\partial \mathbf{p}_1} - \frac{\partial}{\partial \mathbf{p}_2}\right)\right] f_2 = -\int \left(\mathbf{K}_{13} \cdot \frac{\partial}{\partial \mathbf{p}_1} + \mathbf{K}_{23} \cdot \frac{\partial}{\partial \mathbf{p}_2}\right) f_3 \, \mathrm{d}\mathbf{z}_3 \tag{1.13}$$

For particles which interact over a distance of order a in a system of total volume V, the dimensionless parameter $\lambda = Na^3/V$ measures how dilute the particles are. A dilute gas has $\lambda \ll 1$; a quick estimate shows that air at room temperature has $\lambda \sim 10^{-5}$; water at room temperature has $\lambda \sim 10^{-1}$. In either case, λ is a sufficiently small parameter that the distribution functions may be expanded:

$$f_1 = \lambda f_1^0 + \lambda^2 f_1^1 + \lambda^3 f_1^2 + \ldots \tag{1.14}$$
$$f_2 = \lambda^2 f_2^0 + \lambda^3 f_2^1 + \lambda^4 f_2^2 + \ldots \tag{1.15}$$
$$\vdots \tag{1.16}$$

f_1 is a measure of density, and therefore at least linear in λ; f_2 measures correlations and two-particle effects and is therefore of order λ^2. Note that f_i^j is of order λ^{i+j}. Collecting the terms of order λ yields

$$\left(\frac{\partial}{\partial t} + \frac{1}{m}\mathbf{p}_1 \cdot \frac{\partial}{\partial \mathbf{x}_1}\right) f_1^0 = 0. \tag{1.17}$$

This is a collisionless Boltzmann equation, representing particles which coast freely through space and never collide with one another. To take account of collisions requires the next order, at which level of approximation the top two BBGKY equations become

$$\left(\frac{\partial}{\partial t} + \frac{1}{m}\mathbf{p}_1 \cdot \frac{\partial}{\partial \mathbf{x}_1}\right) f_1^1 = -\int \mathbf{K}_{12} \cdot \frac{\partial}{\partial \mathbf{p}_1} f_2^0 \, \mathrm{d}\mathbf{z}_2 \tag{1.18}$$

and

$$\left[\frac{\partial}{\partial t} + \frac{1}{m}\mathbf{p}_1 \cdot \frac{\partial}{\partial \mathbf{x}_1} + \frac{1}{m}\mathbf{p}_2 \cdot \frac{\partial}{\partial \mathbf{x}_2} + \mathbf{K}_{12} \cdot \left(\frac{\partial}{\partial \mathbf{p}_1} - \frac{\partial}{\partial \mathbf{p}_2}\right)\right] f_2^0 = 0. \tag{1.19}$$

If equation (1.19) is integrated over the coordinates $\mathbf{z}_2$ then the first term, containing the time derivative, becomes

$$\int \frac{\partial f_2}{\partial t}\, \mathrm{d}\mathbf{z}_2 = \frac{\partial}{\partial t}\int f_2\, \mathrm{d}\mathbf{z}_2 = \frac{1}{N-1}\frac{\partial}{\partial t} f_1. \tag{1.20}$$

In the limit of large, macroscopic systems, $N \to \infty$, and this term becomes small enough to be neglected. The other terms substitute into (1.18) to express the momentum derivative on the right hand side in terms of spatial derivatives.

At this point, the assumption is made that any two particles in the system will be uncorrelated before and after their brief collision: this is Boltzmann's famous *Stosszahlansatz*, or postulate of molecular chaos (literally, "collision number postulate"). Mathematically, it is equivalent to dropping the two-particle correlation function, and writing the two-particle distribution function as a product of one-particle distribution functions:

$$f_2(\mathbf{z}_1, \mathbf{z_2}) \simeq f_1(\mathbf{z}_1) f_1(\mathbf{z}_2). \tag{1.21}$$

Making the further assumption that the distribution function shows little spatial variation over the range of interaction, and rewriting in terms of the relative velocity $\mathbf{g} = (\mathbf{p}_1 - \mathbf{p}_2)/m$ and relative position $\mathbf{r} = \mathbf{x}_2 - \mathbf{x}_1$:

$$\left(\frac{\partial}{\partial t} + \frac{1}{m}\mathbf{p}_1 \cdot \frac{\partial}{\partial \mathbf{x}_1}\right) f_1 = \int \mathbf{g} \cdot \frac{\partial}{\partial \mathbf{r}} \left[f_1(\mathbf{r}, \mathbf{p}_1) f_1(\mathbf{r}, \mathbf{p}_2)\right]\, \mathrm{d}\mathbf{z}_2 \tag{1.22}$$

Choosing a coordinate system with z-coordinate parallel to the relative velocity $\mathbf{g}$ and impact parameter $\mathbf{b}$ perpendicular to $\mathbf{z}$ finally gives the Boltzmann equation:

$$\left(\frac{\partial}{\partial t} + \frac{1}{m}\mathbf{p}_1 \cdot \frac{\partial}{\partial \mathbf{x}_1}\right) f_1 = \int g \left[f_1(\mathbf{r}, \mathbf{c}_1') f_1(\mathbf{r}, \mathbf{c}_2') - f_1(\mathbf{r}, \mathbf{c}_1) f_1(\mathbf{r}, \mathbf{c}_2)\right]\, \mathrm{d}b\, \mathrm{d}\mathbf{c}_2, \tag{1.23}$$

where the velocities $\mathbf{c}_i = \mathbf{p}_i/m$; primed velocities are post-collisional, and unprimed pre-collisional. The left-hand side of the equation represents changes in the distribution function due to free streaming of particles, and the right hand side — usually called the collision integral Ω — represents changes due to interactions between particles. For brevity, this is written using the Einstein summation convention:

$$(\partial_t + c_\alpha \partial_\alpha) f = \Omega[f] \tag{1.24}$$

The Boltzmann equation is a closed integro-differential equation in the single particle distribution function. It was first written down in 1872 by Boltzmann, who, lacking the BBGKY framework which would only appear several decades after his death, derived it from first principles by considering the kinematics of two-particle collisions and using the Stosszahlansatz to simplify them.

The Boltzmann equation can be considered a mesoscopic description of a fluid: it does not contain the details of every single fluid particle, but it contains more information than just the macroscopic variables. Indeed, the macroscopic quantities such as density ρ, velocity $\mathbf{u}$, and total kinetic energy E are found by integrating out

the "kinetic" degrees of freedom (*i.e.* the information about the velocity distribution of the particles) contained in the distribution function:

$$\rho = m \int f \, \mathrm{d}\mathbf{c} \tag{1.25}$$

$$\rho \mathbf{u} = m \int f \mathbf{c} \, \mathrm{d}\mathbf{c} \tag{1.26}$$

$$E = \frac{m}{2} \int f c^2 \, \mathrm{d}\mathbf{c} \tag{1.27}$$

Historically, one of the most important properties of the Boltzmann equation is that it is time-asymmetric. The asymmetry is introduced by the molecular chaos postulate (1.21), which drops correlations, and cuts the Boltzmann equation off from the higher levels of the BBGKY hierarchy. Boltzmann provided the most (in)famous example of this asymmetry by defining the quantity

$$H \doteq \int f \ln f \, \mathrm{d}\mathbf{c} \, \mathrm{d}\mathbf{x}. \tag{1.28}$$

and showing [7, 9] that

$$\frac{\mathrm{d}H}{\mathrm{d}t} \leq 0. \tag{1.29}$$

This is Boltzmann's famous H-theorem. Literally, it means that the function $H(t)$ monotonically decreases until the detailed balance condition $f'f'_1 = ff_1$ is satisfied, at which point H takes its constant minimum value. It should be noted that H is left unchanged by the streaming process represented by the left-hand side of the Boltzmann equation; changes in H come about entirely due to the collision process. Physically, the H-theorem demonstrates that the Boltzmann equation is time-irreversible: the distribution function evolves inexorably towards its equilibrium value.

By extremizing H and making use of the conservation of particle number, momentum, and energy, it is possible[7] to show that, at equilibrium, the distribution takes the Maxwell-Boltzmann form

$$\bar{f} = \frac{\rho}{(2\pi\theta)^{d/2}} \exp\left(-\frac{(\mathbf{c}-\mathbf{u})^2}{2\theta}\right) \tag{1.30}$$

for d dimensions, where ρ is the mass density, $\mathbf{u}$ is the macroscopic fluid velocity, and $\theta = kT/m$.

1.2.4 The BGK Equation

The collision integral $\Omega[f]$ on the right hand side of the Boltzmann equation can be cumbersome to evaluate. Bhatnagar, Gross, and Krook[10] suggested a linear approximation to it, in which, during the collision process, the distribution relaxes linearly towards equilibrium at a rate controlled by a constant, τ. This gives the continuum BGK equation,

$$\partial_t f + c_\alpha \partial_\alpha f = -\frac{1}{\tau}\left(f - \bar{f}\right). \tag{1.31}$$

Many of the results shown below, including the derivation of the Navier-Stokes equations, may be found[9, 7] using the full expression for the collision operator. For simplicity and brevity, this treatment will proceed from the BGK equation, which retains most of the important properties of the full integro-differential Boltzmann equation, while reducing it to a more manageable partial differential equation.

Properties of the BGK equation

The first three velocity moments of the equilibrium distribution function are easily shown to be

$$\int \bar{f}\, \mathrm{d}\mathbf{c} = \rho \tag{1.32}$$

$$\int \bar{f} c_\alpha\, \mathrm{d}\mathbf{c} = \rho u_\alpha \tag{1.33}$$

$$\int \bar{f} c_\alpha c_\beta\, \mathrm{d}\mathbf{c} = \rho u_\alpha u_\beta + \rho\theta\delta_{\alpha\beta} \tag{1.34}$$

$$\begin{aligned} \int \bar{f} c_\alpha c_\beta c_\gamma &= \rho u_\alpha u_\beta u_\gamma + \rho\theta\left(u_\alpha\delta_{\beta\gamma} + u_\beta\delta_{\alpha\gamma} + u_\gamma\delta_{\alpha\beta}\right) \\ &= \rho u_\alpha u_\beta u_\gamma + \rho\theta\Gamma_{\alpha\beta\gamma} \end{aligned} \tag{1.35}$$

Integrating the BGK equation (1.31) over all velocities and using the moments derived above shows that mass is conserved,

$$\partial_t \rho + c_\alpha \partial_\alpha \rho u_\alpha = 0. \tag{1.36}$$

Taking the first moment with respect to velocity yields

$$\partial_\alpha \rho u_\alpha + \partial_\beta P_{\alpha\beta} = 0. \tag{1.37}$$

This represents conservation of momentum, with the momentum flux tensor $P_{\alpha\beta}$ defined as

$$P_{\alpha\beta} \doteq \int f c_\alpha c_\beta\, \mathrm{d}\mathbf{c}. \tag{1.38}$$

1.2.5 The Navier-Stokes Equations

There are several methods available for coarse-graining the Boltzmann equation to produce hydrodynamic equations. The method of Grad[7] involves expansion of the distribution function in Hermite tensor polynomials; a popular alternative is to use the Chapman-Enskog approximation method.

In the Chapman-Enskog procedure, the time derivative operator is written as a sum of separate operators corresponding to different timescales: $\partial_t = \epsilon\partial_{1t} + \epsilon^2\partial_{2t} + \epsilon^3\partial_{3t} + \ldots$, where ∂_{1t} corresponds to the fastest timescale, ∂_{2t} to the next fastest, and so on; ϵ can be regarded as a bookkeeping parameter, although it is sometimes identified with the Knudsen number. Spatial gradients are coupled to the fastest timescale by writing $\partial_\alpha = \epsilon\partial_\alpha$, and the distribution function is also written as a sum of components due to different timescales:

$$f = f^{(0)} + \epsilon f^{(1)} + \epsilon^2 f^{(2)} + O(\epsilon^3) \tag{1.39}$$

Inserting these expansions into the continuum BGK equation (1.31). and collecting terms with the same power of ϵ yields

$$\begin{aligned} & \epsilon \left[\left(\partial_{1t} + c_\alpha \right) f^{(0)} \right] && (1.40) \\ + \quad & \epsilon^2 \left[\partial_{2t} f^{(0)} + \left(\partial_{1t} + c_\alpha \partial_\alpha \right) f^{(1)} \right] && (1.41) \\ = \quad - \quad & \frac{1}{\tau} \left[f^{(0)} - \bar{f} + \epsilon f^{(1)} + \epsilon^2 f^{(2)} \right] + O(\epsilon^3) && (1.42) \end{aligned}$$

If it is required that the resulting expansion should be true for each timescale separately, then coefficients of individual powers of ϵ may be equated, to give

$$\begin{aligned} f^{(0)} &= \bar{f} && (1.43) \\ -\frac{1}{\tau} f^{(1)} &= \left(\partial_{1t} + c_\alpha \partial_\alpha \right) f^{(0)} && (1.44) \\ -\frac{1}{\tau} f^{(2)} &= \partial_{2t} f^{(0)} + \left(\partial_{1t} + c_\alpha \partial_\alpha \right) f^{(1)} && (1.45) \end{aligned}$$

For conservation of mass and momentum it is required that, for $n > 0$,

$$\begin{aligned} \int f^{(n)} \, \mathrm{d}\mathbf{c} &= 0 && (1.46) \\ \int f^{(n)} c_\alpha \, \mathrm{d}\mathbf{c} &= 0 && (1.47) \end{aligned}$$

Integrating the first-order equation (1.44) over all possible velocities $\mathbf{c}$ yields the continuity equation for particle number density:

$$\partial_{1t} \rho + \partial_\alpha \left(\rho u_\alpha \right) = 0 \qquad (1.48)$$

Multiplying by c_α and integrating gives an equation for the time evolution of the momentum,

$$\rho \partial_{1t} u_\alpha + \rho u_\beta \partial_\beta u_\alpha + \partial_\beta \left(\rho \theta \delta_{\alpha\beta} \right) = 0. \qquad (1.49)$$

This is the Euler equation, representing the motion of an inviscid fluid: momentum dissipation is only treated at second order and beyond. Integrating equation (1.45) over all velocities shows that $\partial_{2t} \rho = 0$. Treatment of the second order momentum equation requires knowledge of the integral

$$\begin{aligned} \frac{-1}{\tau} \int f^{(1)} c_\alpha c_\beta \, \mathrm{d}\mathbf{c} &= \int \left(\partial_{1t} + c_\gamma \partial_\gamma \right) \bar{f} c_\alpha c_\beta \, \mathrm{d}\mathbf{c} \\ &= \partial_{1t} \left(\rho u_\alpha u_\beta + \rho \theta \delta_{\alpha\beta} \right) + \partial_\gamma \left(\rho u_\alpha u_\beta u_\gamma + \rho \theta \Gamma_{\alpha\beta\gamma} \right). \quad (1.50) \end{aligned}$$

Some tedious algebra shows that

$$\partial_\gamma \left(\rho u_\alpha u_\beta u_\gamma \right) + \partial_{1t} \left(\rho u_\alpha u_\beta + \rho \theta \delta_{\alpha\beta} \right) = -\Gamma_{\alpha\beta\gamma} \partial_\gamma \left(\rho \theta \right) - \delta_{\alpha\beta} \rho \theta \partial_\gamma u_\gamma. \qquad (1.51)$$

giving

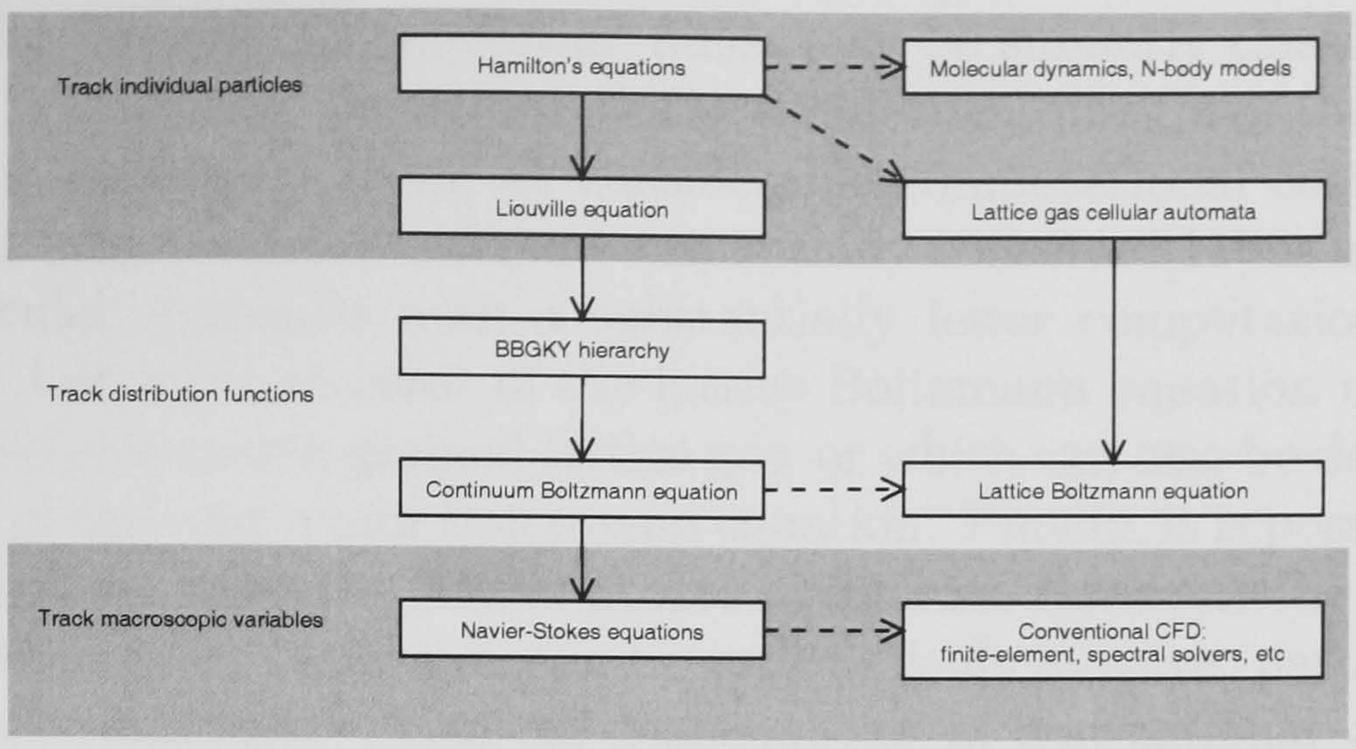


Figure 1.1: Levels of description in classical and computational kinetic theory

$$\frac{-1}{\tau}\int f^{(1)}c_\alpha c_\beta \,\mathrm{d}\mathbf{c} = \rho\theta\left(\partial_\alpha u_\beta + \partial_\beta u_\alpha\right). \tag{1.52}$$

Substituting this into the expression for the first moment of the second-order momentum equation yields

$$\partial_t \rho u_\alpha + \partial_\beta\left(\rho u_\alpha u_\beta + \rho\theta\delta_{\alpha\beta}\right) = \tau\partial_\beta\rho\theta\left(\partial_\alpha u_\beta + \partial_\beta u_\alpha\right). \tag{1.53}$$

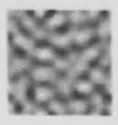
Assuming that the fluid is incompressible simplifies the derivatives to give the appropriate Navier-Stokes equation, with shear viscosity equal to $\tau\theta$.

$$\partial_t u_\alpha + u_\beta\partial_\beta u_\alpha + \frac{1}{\rho}\partial_\beta\rho\theta\delta_{\alpha\beta} = \tau\theta\partial_\beta\partial_\beta u_\alpha. \tag{1.54}$$

It has thus been shown that the microscopically reversible dynamics of a gas of particles obeying Newton's laws will, on a sufficiently coarse-grained scale, give rise to the correct irreversible macroscopic continuum behaviour. It is important to note that, while the Boltzmann equation was originally derived in order to model the behaviour of a gas, solutions to the Boltzmann equation show macroscopic behaviour which obeys the Navier-Stokes equations, which are applicable to fluids in general, including both gases and liquids. While the Boltzmann-level description is, by definition, insufficient to reproduce correlation functions and therefore many of the thermodynamic properties of a liquid, it is still sufficient to reproduce the flow behaviour.

1.3 Computational kinetic theory

While classical kinetic theory spans the scales from atomistic to continuum, the equations at each scale are usually very difficult to solve analytically for anything but the most simple of problems, and one must often resort to numerical solution. The levels of detail described above can be broadly grouped into three types: those which keep track of individual particles, solving Newton's equations or Hamilton's equations for their evolution; those which keep track of a distribution function, such as the Boltzmann equation; and those which only deal with high-level macroscopic variables, such as the Navier-Stokes approach.



Methods for numerically simulating fluids may be similarly classified, as shown in Figure 1.1: Molecular Dynamics (MD) simulations integrate the equations of motion of each individual atom, at considerable computational cost. The lattice gas cellular automaton (LGA) method was initially conceived[11] as a sort of highly stylized molecular dynamics with a substantially lower computational cost: this method is the historical ancestor of the lattice Boltzmann equation method, which can be regarded as a coarse-grained lattice gas, or which can also be derived through discretization of the continuum Boltzmann equation. Finally, it is possible in certain cases to numerically solve the Navier-Stokes equations. However, the Navier-Stokes equations, being highly nonlinear, can be very difficult to solve, particularly under turbulent flow conditions. Perceived inadequacies in conventional computational fluid dynamics (CFD) drove development of novel methods from the late 1980s onwards.

1.3.1 Lattice Gas Automata

In 1986, two papers were published — one by Frisch, Hasslacher, and Pomeau[12], and the other by Wolfram[11] — which both suggested a new method for simulating fluid flow: this method has the substantial advantage that it is unconditionally numerically stable, in contrast to conventional CFD techniques, which can be plagued by instabilities. In addition, it uses only Boolean integer operations, and is thus substantially faster and simpler to implement than systems requiring floating-point logic, and the locality of operations and simplicity of the lattice make it ideally suited to parallel computers.

A lattice gas consists of a set of particles which are constrained to lie on the sites of a regular lattice. Each point $\mathbf{r}$ on the lattice has a set of vectors $\mathbf{c}_i$ connecting it to its nearest neighbours; the velocity of each fluid particle is constrained to be one of these vectors.

During each timestep of the algorithm, lattice gas particles move to adjacent points on the lattice according to their velocity vectors; they then collide with other particles at the same site in such a way as to conserve momentum. For two indistinguishable particles colliding on the hexagonal lattice used by Frisch, Hasslacher and Pomeau — often referred to as the FHP lattice — there are three possible outcomes of a collision which preserve the initial momentum: the outcome is chosen randomly from these possibilities, with equal weights. Most lattice gas models impose an exclusion principle, so that only one particle at a given site could have a given velocity vector at any time, giving rise to Fermi-Dirac statistics for the lattice gas particles.

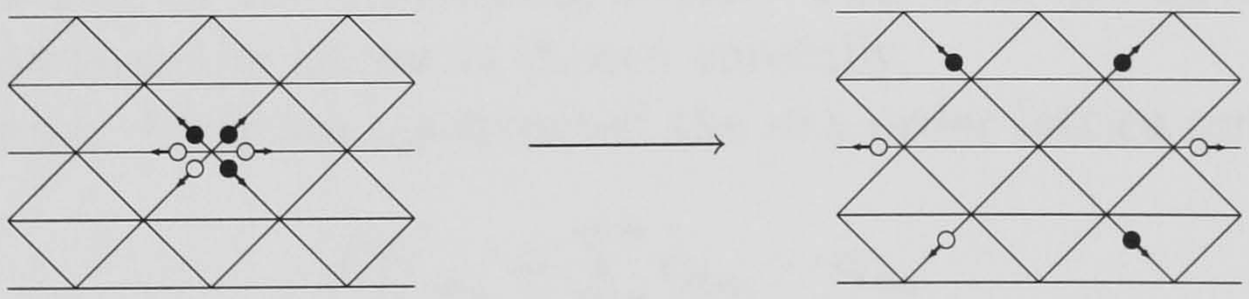


Figure 1.2: The Advection step: particles move along the lattice.

The discrete nature of the lattice gas model makes it a little awkward to treat analytically; the first step in most analytical treatments of LGA is to describe the algorithm in terms of the single-particle distribution function $f_i(\mathbf{r}, t)$, equivalent to

Figure 1.3: The Collision step: particles at a site are redistributed.

the mean number of particles at position $\mathbf{r}$ with velocity $\mathbf{c}_i$. Note that f_i is a real number, in contrast to the integer occupation number n_i of the lattice gas. A lattice equivalent of the Boltzmann equation can then be found:

$$f_i(\mathbf{r} + \mathbf{c}_i, t + 1) - f_i(\mathbf{r}, t) = \Omega_i(\mathbf{r}, t) \tag{1.55}$$

The left-hand side of the equation represents the advection step of the algorithm; the collision step is represented by Ω_i, equivalent to the collision integral in the continuum Boltzmann equation.

The mean density ρ and velocity $\mathbf{u}$ of the lattice gas particles can be found from the distribution function through the relationships:

$$\rho = m \sum_i f_i \tag{1.56}$$

$$\rho u_\alpha = m \sum_i f_i c_{i\alpha} \tag{1.57}$$

Macroscopic equations can then be recovered for LGA by performing a Chapman Enskog procedure, similar to that which produces macroscopic equations from the continuum Boltzmann equation[9, 11, 12].

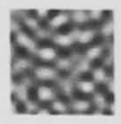
The Navier-Stokes level description of the lattice-gas has the problem that the convection term $(\mathbf{v} \cdot \nabla)\,\mathbf{v}$ is multiplied by a density-dependent term $g(\rho)$, giving the model an unphysical lack of Galilean invariance. In addition, when the term $g(\rho)$ is not equal to unity, the model has a velocity-dependent pressure. A standard workaround for the lack of Galilean invariance is to re-scale the velocity by a factor of $1/g$; the macroscopic equations in terms of the rescaled velocity are Galilean invariant at constant density.[12]

Perhaps the most surprising result of the lattice gas models is that it is possible to describe isotropic hydrodynamics on a regular lattice — it would not be unreasonable to expect that the macroscopic lattice gas fluid should tend to flow in certain specific directions determined by the underlying lattice. This does not have to be the case, however, provided that the lattice is chosen carefully.

In his first paper, Wolfram constructed the nth order lattice tensor:

$$T^{(n)}_{\alpha_1 \ldots \alpha_n} \doteq \sum_i c_{i\alpha_1} \cdots c_{i\alpha_n} \tag{1.58}$$

By performing a Chapman-Enskog procedure on the lattice gas Boltzmann equation, he showed that isotropic hydrodynamics would result if the underlying lattice is isotropic to fourth order — that is to say,



$$T_{\alpha}^{(1)} = 0 \tag{1.59}$$
$$T_{\alpha\beta}^{(2)} = c_{\mathrm{s}}^2 \delta_{\alpha\beta} \tag{1.60}$$
$$T_{\alpha\beta\gamma}^{(3)} = 0 \tag{1.61}$$
$$T_{\alpha\beta\gamma\delta}^{(4)} = c_{\mathrm{s}}^4 \left(\delta_{\alpha\beta}\delta_{\gamma\delta} + \delta_{\alpha\delta}\delta_{\beta\gamma} + \delta_{\alpha\gamma}\delta_{\beta\delta}\right) \tag{1.62}$$

If the lattice obeys these constraints, then lattice-effects will wash out on the macroscopic scale, and the fluid will have no preferred directions. The simplest two-dimensional lattice obeying these constraints is the hexagonal FHP lattice; although no fourth-order isotropic three-dimensional regular lattice exists, it is possible to perform a lattice-gas simulation using a four-dimensional face-centred hypercubic lattice (the "FCHC" lattice), and project it down to three dimensions.

It is notable that some three-dimensional implementations of LGA[13] perform the collisions independently on three sublattices of FCHC for performance reasons, leading to independent conservation of momentum on each sublattice. It is not known whether this leads to any unphysical effects (indeed, this effect does not seem to have been mentioned in the literature), but similar spurious conservation laws can plague discrete fluid models[14].

Since the LGA algorithm is concerned with tracking a finite number of discrete particles around a lattice, and is based on Boolean operations, rather than any floating-point numerical calculation, it is unconditionally numerically stable. In addition, an H-theorem was proved by Hénon[15], showing that the Boltzmann equation describing the evolution of the occupation probabilities for each site on the lattice would inexorably evolve towards an equilibrium state.

It should be clearly understood that the lattice gas particles do not represent molecules in any strictly-defined sense, but mesoscopic "packets" of fluid. Such vagueness in the interpretation can be justified by the fact that the system of interest (the Navier-Stokes equations) is reproduced at the coarse-grained level: in this sense, the LGA approach could be seen as inventing a toy model designed to be as simple as possible, whose lack of connection with lower levels of description (such as real molecules) is justified by correct high-level behaviour. Some effort has been invested[16] in trying to find the simplest possible lattice gas which still produces Navier-Stokes behaviour.

1.3.2 The Lattice Boltzmann Equation

Although a Boltzmann equation formulation of LGA had often been used to treat the model analytically, it was not until 1988 that McNamara and Zanetti[17] suggested solving the lattice Boltzmann equation directly, and computing successive values of the real-valued single-particle distribution function $f_i(\mathbf{r}, t)$, rather than finding its value through computationally-intensive averaging of a lattice gas model. McNamara and Zanetti's model used a collision operator derived from a variant of the FHP lattice gas described by d'Humières and Lallemand[18]. The collision operator contained polynomials in f_i with as many terms as there were effective collisions; this was sufficiently complicated to work with that the model was never generalized to three dimensions.

Higuera *et al* [19, 20] suggested a method for obtaining simplified linear collision operators, which could be generalised to three dimensions. However, these models

were still plagued by the density-dependent factor $g(\rho)$ seen in LGA.

1.3.3 Lattice BGK

That the collision term in lattice Boltzmann models was the cause of difficulties should not be surprising; the collision integral in the continuum lattice Boltzmann equation is not particularly easy to work with either.

However, an H-theorem exists for the continuum Boltzmann equation, requiring the single-particle distribution function $f(\mathbf{r}, t)$ it describes to evolve towards a well-defined equilibrium value $f^{(\mathrm{eq})}(\mathbf{r}, t)$. In the mid 1950s, Bhatnagar, Gross, and Krook[10] had taken advantage of this to suggest that a reasonable approximation to the continuum collision integral Ω would be a term describing the linear relaxation of the distribution function towards its equilibrium value, at a rate controlled by a single relaxation time parameter, τ :

$$\Omega \simeq \frac{f^{(\mathrm{eq})} - f}{\tau}. \tag{1.63}$$

An analogous modification to the lattice Boltzmann equation was performed independently by Qian *et al* [21], and Chen *et al*[22, 23], approximating the collision operator to produce what is now commonly referred to as the "lattice BGK equation":

$$f_i(\mathbf{r} + \mathbf{c}_i, t + 1) - f_i(\mathbf{r}, t) = -\frac{1}{\tau}\left[f_i(\mathbf{r}, t) - \bar{f}_i(\mathbf{r}, t)\right] \tag{1.64}$$

In continuum Boltzmann equation treatments, the equilibrium distribution can be found from the lower bound of the H-theorem. In introducing the BGK approximation, Qian *et al* simply used the equilibrium distribution for the Boltzmann equation describing a lattice gas; note, however, that this is *not* necessarily the thermodynamic equilibrium distribution function for a lattice Boltzmann model, which is a different dynamical system.

This is a point which was not made terribly clear in the early literature surrounding lattice BGK — the "local equilibrium" value $\bar{f}_i$ towards which the distribution function f_i relaxes is an equilibrium value only in the sense that if f_i ever takes the exact value $\bar{f}_i$ at all points on the lattice, then the system will cease to evolve in time. It can be shown[24] that systems (such as most lattice BGK models) whose equilibrium distribution is a polynomial function of macroscopic variables cannot possess an H-theorem: in such systems, there is no universal guarantee that f_i will ever attain the value $\bar{f}_i$.

Qian also developed a method for making certain anisotropic lattices produce isotropic hydrodynamics, by introducing a weighting factor T_i for each lattice vector $\mathbf{c}_i$, with the result that the lattice tensors are redefined to be

$$T^{(n)}_{\alpha_1 \ldots \alpha_n} \doteq \sum_i T_i c_{i\alpha_1} \cdots c_{i\alpha_n} \tag{1.65}$$

For the lattices enumerated in the paper by Qian, d'Humières and Lallemand[21], the redefined lattice tensor (1.65) obeys the isotropy requirements (1.59–1.62).

The lattice gas Boltzmann equation equilibrium distribution $\bar{f}_i$ for these lattices was found[25] to be given in terms of the velocity $\mathbf{u}$ and number density $n = \rho/m$ of particles at a site, by

$$\bar{f}_i = nT_i \left[1 + \frac{c_{i\alpha}u_\alpha}{c_s^2} + \frac{u_\alpha u_\beta}{2c_s^2} \left(\frac{c_{i\alpha}c_{i\beta}}{c_s^2} - \delta_{\alpha\beta} \right) \right] \tag{1.66}$$

By multiplying and summing over lattice vectors according to the relations (1.56.1.57). and using the property of lattice isotropy (1.59–1.62) to simplify the lattice tensors, it can easily be seen that this equilibrium distribution conserves the mass and momentum of colliding particles:

$$\sum_i \bar{f}_i = n \left[\left(1 - \frac{1}{2c_s^2} u_\alpha u_\beta \right) \sum_i T_i + \frac{1}{c_s^2} u_\alpha T_\alpha^{(1)} + \frac{1}{2c_s^4} u_\alpha u_\beta T_{\alpha\beta}^{(2)} \right] = n \tag{1.67}$$

$$\sum_i \bar{f}_i c_{i\alpha} = n \left[\left(1 - \frac{1}{2c_s^2} u_\alpha u_\beta \right) T_\alpha^{(1)} + \frac{1}{c_s^2} u_\alpha T_{\alpha\beta}^{(2)} + \frac{1}{2c_s^4} u_\alpha u_\beta T_{\alpha\beta\gamma}^{(3)} \right] = n u_\alpha \tag{1.68}$$

Note the additional requirement that $\sum_i T_i = 1$.

The resulting "lattice BGK" model has substantial improvements over lattice gas: it is less noisy, and therefore requires much less computationally-intensive averaging; it is much simpler to implement, since the algorithm is deterministic and does not require the use of various computational tricks to speed up the process of choosing the outcome of collisions. Moreover, the lattice BGK model described above is Galilean invariant, and does not have a velocity-dependent pressure, which were two considerable problems with lattice gases.

However, these advantages were obtained at the cost of going back to requiring the use of floating-point arithmetic, and at the cost of numerical instabilities reappearing in the model: unlike lattice gas, lattice BGK is not unconditionally stable.

A factor contributing to the resulting popularity of the LBGK method is that the introduction of extra terms representing eternal forces or interactions between fluid particles is much easier than in lattice gas models. Several systems for introducing forcing into LBGK have been proposed, some of which contain a single term representing the force on a unit particle, which can then be set to whatever value is required. This ease is in contrast to lattice gas methods, where one is required to either shift the probability distribution of collision outcomes[26], or repopulate the lattice vectors at each timestep in such a way as to increase the nett momentum of the particles.[27]

It has more recently been shown, for example by He and Luo[28, 29] and Abe[30] that the lattice BGK method may also be derived through a systematic discretization of the continuum Boltzmann BGK equation, completely independently of lattice-gas automata.

1.4 Complex fluid modelling

The work described so far allows one to model the behaviour of fluids of various sorts. The lattice Boltzmann method, which can be regarded as a discretized form of the continuum Boltzmann equation, offers a way of deriving the macroscopic flow behaviour of a simple fluid while still retaining some level of kinetic description: the

simplicity and locality of the method permit fast implementations on modern computing hardware, and are particularly well suited to parallel computing[31]. Complex fluids, as stated at the outset, require more than just hydrodynamics: however, the mesoscale nature of the lattice Boltzmann method facilitates its extension to cover more sophisticated systems.

There are other techniques currently in use for mesoscale modelling besides lattice Boltzmann, although they all typically take the form of hydrodynamics plus some extra parts to represent complex fluids.

1.4.1 Dissipative Particle Dynamics

The Dissipative Particle Dynamics (DPD) model, essentially a Langevin equation with momentum conservation, was originally proposed by Hoogerbrugge and Koelman [32], and models the behaviour of fictitious mesoscopic "particles", each corresponding to a large number of microscopic fluid particles; the original formulation was modified slightly by Español and Warren[33], so that the model produces a Gibbsian equilibrium distribution.

The state of a DPD system is completely specified by the positions $\mathbf{r}_i$, masses m_i, and momenta $\mathbf{p}_i$ of all N constituent dissipative particles; the position and momenta are continuous, but the model uses discrete timesteps δt; each timestep consists of a collision stage and an advection stage.

In the propagation step, particles coast along according to their momentum vectors, giving

$$\dot{\mathbf{r}}_i = \frac{1}{m}\mathbf{p}_i \tag{1.69}$$

Following the notation of Español and Warren, the particle momenta are updated according to the rule:

$$\dot{\mathbf{p}}_i = \sum_{j\neq i}\mathbf{F}_{ij}^C + \sum_{j\neq i}\mathbf{F}_{ij}^D + \sum_{j\neq i}\mathbf{F}_{ij}^R \tag{1.70}$$

$\mathbf{F}_{ij}^C$ is a conservative interaction force between the dissipative particles, $\mathbf{F}_{ij}^D$ is a dissipative, viscous force, and $\mathbf{F}_{ij}^R$ is a random force; stochastic fluctuations are introduced into the model through the last term. A simple form of these forces, satisfying Galilean invariance and isotropy, and simplifying further theoretical treatment of the system, is

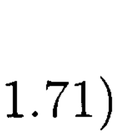
$$\mathbf{F}_{ij}^D = -\gamma\omega_D(r_{ij})(\mathbf{e}_{ij}\cdot\mathbf{v}_{ij})\mathbf{e}_{ij} \tag{1.71}$$

$$\mathbf{F}_{ij}^R = \sigma\omega_R(r_{ij})\mathbf{e}_{ij}\zeta_{ij} \tag{1.72}$$

where $\mathbf{e}_{ij}$ is the unit vector pointing from the jth particle to the ith particle, $\mathbf{v}_{ij}$ is the relative velocity of the particles, and ζ_{ij} is a Gaussian white noise term such that $\zeta_{ij} = \zeta_{ji}$, and with the properties that

$$\begin{aligned}\langle\zeta_{ij}(t)\rangle &= 0\\ \langle\zeta_{ij}(t)\zeta_{i'j'}(t')\rangle &= (\delta_{ii'}\delta_{jj'} + \delta_{ij'}\delta_{ji'})\delta(t-t')\end{aligned} \tag{1.73}$$

The weighting functions ω_D and ω_R define the range of interaction for the dissipative and random forces; typically, there is no interaction beyond some cutoff



radius r_c — localizing interactions in this way makes the algorithm more computationally efficient. γ, which controls the magnitude of the dissipative force. can be regarded as a frictional coefficient, and σ, controlling the amplitude of the noise, is related to the temperature k_BT of the system:

$$\sigma = (2k_BT\gamma)^{\frac{1}{2}} \tag{1.74}$$

This formulation of DPD has been shown[34] to satisfy the criterion of detailed balance, namely that the product of the transition probability of the system from a state A to another state B and the probability of obtaining state A is equal to the product of the transition probability for $B \rightarrow A$ and the probability of B:

$$W(A \rightarrow B)P(A,t) = W(B \rightarrow A)P(B,t) \tag{1.75}$$

Detailed balance can be shown to be a sufficient, but not necessary, condition for a Gibbsian equilibrium condition to exist[35]. Furthermore, an H-theorem was derived for single-component DPD[36], showing that the system will inevitably equilibrate to a Gibbsian distribution.

The DPD algorithm has been extended to cover multicomponent fluids; Coveney and Español [37], and Marsh and Coveney[38] derived detailed balance conditions and H-theorems for such models.

DPD has been used to model the separation of immiscible fluids[39], phase separation under shear[40], formation of various amphiphilic phases[41], and colloidal suspensions[42]. More recently, Español [43] proposed a variant of DPD, where the dissipative particles are regarded as cells in a Voronoi lattice, with variable masses and sizes, rather than fixed spheres; Flekkøy *et al* [44] showed that this model may be derived from Molecular Dynamics through a systematic coarse-graining procedure. It should be noted, however, that the pairwise interactions, amongst other things, make the method slow in comparison to lattice Boltzmann.

1.4.2 The method of Malevanets and Kapral

Malevanets and Kapral[45] proposed a model which seems to have almost as many names as it does practitioners: real-coded lattice gas[46, 47], stochastic rotation dynamics[48], discrete simulation automaton, or simply "Malevanets-Kapral". The fluid to be simulated is divided into particles whose positions occupy sites on a regular cubic lattice; each particle has a real-valued velocity $\mathbf{v}_i$. The particles are stochastically streamed to other points on the lattice, in such a way that the expectation value of a particle's displacement is given by its velocity vector.

A collision stage then takes place, redistributing the velocities of the particles at each site on the lattice. At each site, the centre of mass velocity $\mathbf{V}$ is found; the particle velocities are then transformed according to a random rotation matrix σ, which differs from site to site.

$$\mathbf{v}_i' = \mathbf{V} + \sigma(\mathbf{v}_i - \mathbf{V}) \tag{1.76}$$

This collision step can be regarded as rotating all the velocity vectors at a site around the centre-of-mass velocity, which therefore remains unchanged. Hence, the total momentum at each site remains unchanged, producing hydrodynamic behaviour, while the randomization of the velocities produces dissipative behaviour. It was also shown[49] that an H-theorem exists for this model.

Hashimoto, Chen, and Ohashi[46] extended the real-coded lattice gas technique to model binary immiscible fluids, by adjusting the rotation matrix σ for each different phase, in such a way as to make the phases repel: it was verified that this model obeyed Laplace's law, namely that the pressure difference between the inside and the outside of a droplet is inversely proportional to the droplet radius. Sakai *et al* [47] extended this model to include an amphiphilic surfactant phase, and successfully modelled the process of micelle formation.

1.4.3 Hybrid methods

One approach to tackling mesoscale problems is to use intermediate-level descriptions as detailed above. Another approach which has been receiving increased attention is the coupling of two different algorithms: one to deal with macroscopic aspects of the system such as hydrodynamic effects, and the other to deal with microscopic effects. For example, an interesting hybrid model was developed by Malevanets and Yeomans[50], where Molecular Dynamics techniques were used to model the behaviour of a short polymer chain, and a real-coded lattice gas was coupled to the MD model to add hydrodynamic effects. Delgado-Buscalioni and Coveney[51] describe a scheme for coupling molecular dynamics calculations to continuum fluid dynamics; considerable care must be taken to ensure consistent behaviour at the interface. While lattice Boltzmann on its own is incapable of dealing with the folding of polymer chains or behaviour of suspended particles, it can be used to provide the hydrodynamic part of hybrid models[52, 53].

1.4.4 Complex fluids in Lattice Boltzmann

There are several popular ways of extending lattice Boltzmann to cover complex fluids. Mixtures of fluids which interact in some way have received particular interest, since they are easy to examine in this framework, and correspond to many situations of interest in the real world, such as salad dressings, polymer blends, or the oil-water mixtures in oil wells.

...from the bottom up

One approach, often described as the "bottom-up" , or emergent approach, is to invent or postulate a microscopic interaction model (such as a mean-field force between particles), which produces coarse-grained behaviour similar to that seen in real life. Bottom-up approaches have the advantage that they permit the description of systems whose macroscopic description is too complicated to simply write down. In cases with the invention of fictitious physics, a system may be boiled down to its absolute minimum level of detail required to reproduce the macroscopic behaviour, allowing the examination of how such behaviour comes about. Derivation of any kind of continuum description from such models. even at equilibrium, may still remain difficult.

The Rothman-Keller LBE Model

The Rothman-Keller model for immiscible lattice gases altered the probabilities of different collision outcomes to favour those which would bring about a separation of the fluid components. Gunstensen *et al* [54] suggested a lattice Boltz-

mann model based around the linearized matrix collision operator of McNamara and Zanetti[17]: specifically, they added a perturbation to the collision operator to produce an anisotropic pressure tensor near interfaces between two immiscible fluids while still conserving the total momentum; the outgoing collided particles were then recoloured to produce zero diffusivity of one colour into the other.

Laplace's law was verified for this model, and simulations of capillary waves were carried out. However, the model used the inefficient matrix collision operator, and could not handle two fluids with different densities and viscosities.[55]

Grunau *et al* [56] modified this model to take the BGK form, in a way that permitted variable densities and viscosities. The collision step still consisted of two stages: the first collided the particles according to the BGK operator(1.63); the second recoloured the outgoing particles in such a way as to maximize the outgoing colour flux, defined as

$$\mathbf{\Phi}(\mathbf{r}) = \sum_i \left[\rho^r(\mathbf{r}+\mathbf{c}_i) - \rho^b(\mathbf{r}+\mathbf{c}_i)\right]\mathbf{c}_i \tag{1.77}$$

The kinematic viscosity $\nu_\sigma$of a given pure phase in this model was a function only of its relaxation time τ_σ, allowing the simulation of fluids with differing viscosities.

The Shan-Chen Model

Shan and Chen[57] suggested another generalization of the lattice BGK model to multicomponent fluids with interactions. The form (1.66) of the equilibrium distribution function is retained, but a different velocity $\mathbf{v}^\sigma$ is used for each component, so that the kinetic equation obeyed by the model is

$$f_i^\sigma(\mathbf{r}+\mathbf{c}_i, t+1) - f_i^\sigma(\mathbf{r},t) = -\frac{1}{\tau_\sigma}\left[f_i^\sigma(\mathbf{r},t) - \bar{f}_i^\sigma(\mathbf{v}^\sigma,t)\right] \tag{1.78}$$

It is assumed that, during each collision step, the velocities of each different phase quickly equalize in the absence of forces. The velocity v_α^σ of component σ used in the equilibrium distribution function is written as a sum of this equalized velocity u'_α, and a small increment due to the force $\mathbf{F}^\sigma$ acting on that component

$$v_\alpha^\sigma \doteq u'_\alpha + \frac{\tau_\sigma}{\rho^\sigma}F_\alpha^\sigma \tag{1.79}$$

The requirements were imposed that the mass of a given component is conserved, and that the change in total momentum at a given site is entirely due to the sum of all forces acting on the fluid mixture at that site. Note that momentum is not conserved locally in this model: adjacent sites exchange momentum during the collision step.

$$m_\sigma \sum_i \Omega_i^\sigma = 0 \tag{1.80}$$

$$\sum_\sigma m_\sigma \sum_i \Omega_i^\sigma c_{i\alpha} = \sum_\sigma F_\alpha^\sigma \tag{1.81}$$

This produces an expression for the equalized velocity u'_α.

$$u'_\alpha \doteq \left(\sum_\sigma \frac{\rho^\sigma}{\tau_\sigma}u_\alpha^\sigma\right) / \left(\sum_\sigma \frac{\rho^\sigma}{\tau_\sigma}\right) \tag{1.82}$$

The force $\mathbf{F}^\sigma$ can take arbitrary values, such as $g\rho^\sigma\hat{\mathbf{z}}$ to give a gravitational force acting in the z-direction. In order to produce nearest-neighbour interactions between components, Shan and Chen suggested a force taking the form

$$F_\alpha^\sigma(\mathbf{x}) = -\psi^\sigma(\mathbf{x})\sum_{\bar{\sigma}} g_{\sigma\bar{\sigma}}\sum_i \psi^{\bar{\sigma}}(\mathbf{x}+\mathbf{c}_i)c_{i\alpha} \tag{1.83}$$

$\psi^\sigma(\mathbf{x}) = \psi^\sigma(\rho^\sigma(\mathbf{x}))$ is an effective mass for component σ; $g_{\sigma\bar{\sigma}}$ is a constant controlling the interaction between two different components.

In order to model immiscible fluids, ψ^σ may simply be set to ρ^σ; $g_{\sigma\bar{\sigma}}$ is set to zero for $\sigma = \bar{\sigma}$, and a positive value for $\sigma \neq \bar{\sigma}$: for a given component, this will produce a force in a direction directly away from any interfaces. Taylor-expanding the second effective-mass term in (1.83) gives

$$\psi^{\bar{\sigma}}(\mathbf{x}+\mathbf{c}_i) \simeq \psi^{\bar{\sigma}}(\mathbf{x}) + c_{i\beta}\partial_\beta\psi^{\bar{\sigma}}(\mathbf{x}) + \frac{1}{2}c_{i\beta}c_{i\gamma}\partial_\beta\partial_\gamma\psi^{\bar{\sigma}}(\mathbf{x}) + \ldots \tag{1.84}$$

Substituting into (1.83), and using the expressions (1.59–1.62) to perform the summation over the lattice vectors $\mathbf{c}_i$ shows that, to second-order, the force acts in a direction opposite to the concentration gradient of the components.

$$\begin{aligned} F_\alpha^\sigma &= \psi^\sigma\sum_{\bar{\sigma}} g_{\sigma\bar{\sigma}}\sum_i \left(c_{i\alpha} + c_{i\alpha}c_{i\beta}\partial_\beta + \frac{1}{2}c_{i\alpha}c_{i\beta}\partial_\beta\partial_\gamma + \ldots\right)\psi^{\bar{\sigma}} \\ &\simeq c_s^2\psi^\sigma\sum_{\bar{\sigma}} g_{\sigma\bar{\sigma}}\partial_\alpha\psi^{\bar{\sigma}} \end{aligned} \tag{1.85}$$

The Shan-Chen model may also be used to describe liquid-gas mixtures: defining a nonzero $g_{\sigma\sigma}$ gives rise to a repulsive force between particles of the same phase[58]. A coexistence curve was determined for a simple liquid-gas mixture, and it was found[58] that this curve agreed with the classic thermodynamic theory if the effective mass function was chosen to be $\psi(\rho) = \psi_0\exp(-\rho_0/\rho)$ for arbitrary constants ψ_0,ρ_0.

. . . from the top down

Another way, the "top-down" approach, is to stipulate that the fluid will obey some specific macroscopic laws, such as extremizing a free-energy functional at equilibrium; the lattice Boltzmann model is then modified accordingly.

The top-down approach has the substantial advantage that the continuum behaviour of the model is known by definition; however, such models may be difficult to formulate, especially in systems which are far from equilibrium, and can also be extremely difficult to solve. Moreover, they give no information about how the microscopic interactions give rise to the macroscopic behaviour, which is simply dialled in.

The Oxford Model

Swift *et al* [59] introduced a multiphase lattice Boltzmann model designed to produce a well-defined isothermal equation of state. They defined a free energy functional of the density n:

$$\Psi[n] = \int \left[\frac{\kappa}{2}|\nabla n(\mathbf{r})|^2 + \upsilon(n(\mathbf{r}))\right] \, \mathrm{d}\mathbf{r} \tag{1.86}$$

The first term, controlled by the constant κ, gives the contribution from any density gradients; the second describes the bulk free energy density. The pressure is defined by

$$p(\mathbf{r}) = n\frac{\delta\Psi}{\delta n} - \Psi = p_0 - \kappa n\nabla^2 n - \frac{\kappa}{2}|\nabla n|^2 \tag{1.87}$$

where $p_0 = n\psi'(n) - \psi(n)$ is the equation of state of the fluid. The equilibrium distribution $\bar{f}_i$ is then constructed to conserve mass and momentum, and to produce a pressure tensor of the form

$$P_{\alpha\beta}(\mathbf{r}) = p(\mathbf{r})\delta_{\alpha\beta} + \kappa\partial_\alpha n\partial_\beta n \tag{1.88}$$

This system has the considerable advantage that it is possible to directly choose parameters such as the surface tension, by choosing an appropriate form of the free energy Ψ, and is easily generalized to multiphase fluids[60].

However, a problem with this model is that the viscosity term in the macroscopic equations is not Galilean invariant[61], an effect which is substantial in the presence of large density gradients, although attempts have been made[62] to correct this. The model came under considerable criticism from Luo[63] on the grounds that, while it postulates a thermodynamically-based interaction scheme, the behaviour of the model is not thermodynamically correct.

...from the continuum Boltzmann equation

The "top-down" method is not entirely top-down, since it starts at the middle with a lattice Boltzmann model, and then adds interactions from the continuum (top) level. A truly top-down approach would be to write down a continuum Boltzmann equation generalized to cover the complex fluid in question, and then, following the approach of d'Humières, Lallemand, and Luo[64, 65] discretize this to obtain a lattice Boltzmann model. This has been done, for example to cover the case of single-component liquid-gas mixtures[29] or binary mixtures[66]. As with the top-down approach, this still requires the existence of a continuum-level description. However, it remains unclear whether or not an interacting-fluid model can be framed purely in the Boltzmann or BGK approximations, particularly if thermohydrodynamics is of concern[67].

1.4.5 Amphiphilic lattice Boltzmann

As with some other mesoscale fluid methods[13], amphiphilic fluids may be treated in the LB framework by introducing a new species of particle with an orientational degree of freedom, as described by Chen *et al* [68]. The particles of this new species are each given a vector dipole moment $\mathbf{d}$ which has maximum magnitude d_0, corresponding to complete alignment of the constituent molecules. This is represented in the model by a dipole field $\mathbf{d}(\mathbf{x}, t)$ representing the average orientation of any amphiphile present at site $\mathbf{x}$. In the advection step, values of $\mathbf{d}(\mathbf{x}, t)$ are propagated around the lattice according to

$$\rho^{\mathrm{s}}(\mathbf{x}, t+1)\mathbf{d}(\mathbf{x}, t+1) = \sum_i \tilde{f}_i^{\mathrm{s}}(\mathbf{x} - \mathbf{c}_i, t)\tilde{\mathbf{d}}(\mathbf{x} - \mathbf{c}_i, t). \tag{1.89}$$

where tildes denote post-collision values. During the collision step itself, the dipole moments evolve in a BGK process controlled by a dipole relaxation time τ_d:

$$\tilde{\mathbf{d}}(\mathbf{x}, t) = \mathbf{d}(\mathbf{x}, t) - \frac{1}{\tau_d}\left[\mathbf{d}(\mathbf{x}, t) - \mathbf{d}^{(\mathrm{eq})}(\mathbf{x}, t)\right]. \tag{1.90}$$

The equilibrium dipole moment $\mathbf{d}^{(\mathrm{eq})}$ is aligned with the colour field $\mathbf{h}$:

$$\mathbf{d}^{(\mathrm{eq})} \simeq \frac{\beta d_0}{3}\mathbf{h} \tag{1.91}$$

The colour field contains a component $\mathbf{h}^c$ due to coloured particles such as oil and water, and a part $\mathbf{h}^s$ due to dipoles. The former can be found from the populations of surrounding lattice sites,

$$\mathbf{h}^c = \sum_\sigma q^\sigma \sum_i \rho^\sigma(\mathbf{x} + \mathbf{c}_i)\mathbf{c}_i, \tag{1.92}$$

where q^σ is a colour charge, such as $+1$ for red particles, -1 for blue particles, and 0 for amphiphile particles. The field due to other dipoles turns out to be given by

$$\mathbf{h}^s(\mathbf{x}, t) = \sum_i \left[\sum_{j\neq 0} f_i^s(\mathbf{x} + \mathbf{c}_i, t)\theta_j \cdot \mathbf{d}_i(\mathbf{x} + \mathbf{c}_j, t) + f_i^s(\mathbf{x}, t)\mathbf{d}_i(\mathbf{x}, t)\right], \tag{1.93}$$

where the second-rank tensor θ_j is defined in terms of the unit tensor $\mathbf{I}$ and lattice vector $\mathbf{c}_j$ as

$$\theta_j = \mathbf{I} - \frac{D}{c^2}\mathbf{c}_j\mathbf{c}_j. \tag{1.94}$$

In the presence of an amphiphilic species, the force on an oil or water particle includes an additional term $\mathbf{F}^{\sigma,s}$ to account for the colour field due to the amphiphiles. By treating an amphiphilic particle as a pair of oil and water particles with a very small separation $\mathbf{d}$, and Taylor-expanding in $\mathbf{d}$, it can be shown that this term is given by

$$\mathbf{F}^{\sigma,s}(\mathbf{x}, t) = -2\psi^\sigma(\mathbf{x}, t)g_{\sigma s}\sum_{i\neq 0}\tilde{\mathbf{d}}(\mathbf{x} + \mathbf{c}_i, t)\cdot\theta_i\psi^s(\mathbf{x} + \mathbf{c}_i, t), \tag{1.95}$$

where $g_{\sigma s}$ is a constant controlling the strength of the interaction between amphiphiles and non-amphiphiles.

While they do not possess a net colour charge, the amphiphiles also experience a force due to the colour field, consisting of a part $\mathbf{F}^{s,c}$ due to ordinary species, and a part $\mathbf{F}^{s,s}$ due to other amphiphiles. These terms are given by

$$\mathbf{F}^{s,c} = 2\psi^s(\mathbf{x}, t)\tilde{\mathbf{d}}(\mathbf{x}, t)\cdot\sum_\sigma g_{\sigma s}\sum_{i\neq 0}\theta_i\psi^\sigma(\mathbf{x} + \mathbf{c}_i, t) \tag{1.96}$$

$$\mathbf{F}^{s,s} = -\frac{4D}{c^2} g_{ss} \psi^s(\mathbf{x}) \sum_i \Big\{ \tilde{\mathbf{d}}(\mathbf{x}+\mathbf{c}_i) \cdot \theta_i \cdot \tilde{\mathbf{d}}(\mathbf{x}) \mathbf{c}_i + \Big[\tilde{\mathbf{d}}(\mathbf{x}+\mathbf{c}_i) \tilde{\mathbf{d}}(\mathbf{x}) + \tilde{\mathbf{d}}(\mathbf{x}) \tilde{\mathbf{d}}(\mathbf{x}+\mathbf{c}_i) \Big] \cdot \mathbf{c}_i \Big\} \psi^s(\mathbf{x}+\mathbf{c}_i). \tag{1.97}$$

To summarize, the interactions between fluid components are governed by the coupling constants g_{cc}, g_{cs}, and g_{ss}, controlling the interaction between different sorts of coloured particles, between coloured particles and amphiphiles, and between the amphiphiles.

While the form of the interactions seems straightforward at a mesoscopic level, it is essentially phenomenological, and it is not necessarily easy to relate the interaction scheme or its coupling constants to either microscopic molecular characteristics, or to macroscopic phase behaviour. Some theoretical progress has been made in relating LGA amphiphile models to an underlying microscopic model [69], although macroscopic behaviour is very sensitive not only to the values of the coupling constants, but to the concentrations of each species present, *inter alia*. Different values of these parameters will give rise to a wide variety of different phases [70], such as spherical and wormlike micelles, sponges, lamellae, or droplets: the phase behaviour can be very difficult to predict beforehand from the simulation parameters, and brute-force parameter searches are often resorted to [13], although this procedure can be streamlined somewhat through the use of computational steering[71].

Chapter 2

Spinodal Decomposition

Two fluids considered immiscible at room temperature will normally mix above some ordering temperature T_c, when the thermal noise swamps the repulsion between the two fluid components. If a mixture of the two fluids above T_c is forced or "quenched" below the ordering temperature, the fluids will separate; the separation process is known as spinodal decomposition. Much of the theory surrounding the process has been summarized by Bray[72] and Furukawa[73]; however, recent work has shown that the process is not as straightforward as these theoretical treatments may suggest.

Spinodal decomposition is an ideal application for lattice Boltzmann, since it is not easily treatable in theory without resorting to very crude approximations, which often neglect either hydrodynamics or the effects of the interfaces between components. It is a process of interest to industry, since phase separation plays an important part in, for example, production of polyurethane foams.

This chapter describes some of the theory of spinodal decomposition, and in particular the different mechanisms which are involved at different stages of the process. Lattice Boltzmann simulations of spinodal decomposition are described, and agreement is found with a free-energy theory of early-time behaviour, and with a scaling analysis of longer-time behaviour. Finally, it is demonstrated that the Shan-Chen LBE model can show breakdown of dynamical scaling, an effect observed in experiments and in other numerical studies.

2.1 Theory of spinodal decomposition

A two-component fluid may be described by a nett fluid velocity $\mathbf{u}(\mathbf{r})$ and the densities $\rho^{\mathrm{A}}(\mathbf{r})$ and $\rho^{\mathrm{B}}(\mathbf{r})$ of each component, at each point $\mathbf{r}$ in the fluid. In the incompressible régime, it can be useful to work with the conserved order parameter $\phi(\mathbf{r}) = \rho^{\mathrm{A}}(\mathbf{r}) - \rho^{\mathrm{B}}(\mathbf{r})$ which describes the degree of separation of the fluids. A site with unit particle density which contains only species A will have $\phi = +1$; a site containing only species B will have $\phi = -1$, and a site with equal proportions will have $\phi = 0$. At very early times, which the system is well-mixed, ϕ will be very small; during the demixing process, the system separates out into regions of $\phi = \pm 1$.

The process of phase separation can be divided into several régimes, each dominated by a different physical process. For many of these régimes, it is thought that the *dynamical scaling hypothesis* holds — that is to say, snapshots of a system at two different times will have identical morphology when each snapshot is scaled by its single characteristic length scale $L(T)$, whose time dependence is of the form

$L(T) \sim T^{\alpha}$.

If it is assumed that the only parameters determining the behaviour of the system are the density ρ, kinematic viscosity ν, and surface tension σ, then only one length scale $L_0 = \rho\nu^2/\sigma$ and one time scale $T_0 = \rho^2\nu^3/\sigma^2$ may be constructed. Lengths L and times T measured in simulations may then be described in terms of the reduced variables $\ell = L/L_0$, and $t = T/T_0$. If dynamical scaling holds for a range of phase-separating systems, then the evolution of reduced domain size ℓ plotted against the reduced time t should collapse onto the same curve for all such systems.

A simple treatment of the process divides it into three main régimes: early-time diffusive growth, during which small density fluctuations grow to form interfaces between domains containing a single phase, a viscous hydrodynamic régime, where the coarsening of these domains is driven by surface-tension effects, and a late-time régime, where coarsening is driven by inertial hydrodynamic flow. Grant and Elder[74] also proposed the existence of a turbulent régime, where the flow is sufficiently strong to remix the fluid, preventing further domain growth. Theoretical treatments[73] suggest that a single length scale should exist at a given time in each of these régimes, and that this length scale varies as a power of the time; most models of spinodal decomposition have attempted to determine these growth exponents. As detailed in the next section, most examinations of spinodal decomposition have found late-time inertial hydrodynamic growth exponents of $\frac{2}{3}$, in agreement with Furukawa's predictions, with early-time exponents of $\frac{1}{2}$ or $\frac{1}{3}$ in 2D and 1 in 3D. Although Grant and Elder proposed that the growth exponent should not exceed $\frac{1}{2}$ for sufficiently high Reynolds numbers, this régime has so far not been seen.

2.1.1 Interface formation

During the very early stages of spinodal decomposition from a deep quench, the order parameter will be very small. Small fluctuations in the order parameter will gradually become larger, as particles of a given component gradually diffuse towards one another, and away from particles of the other component. The fluid velocity will be small, so hydrodynamics may be neglected.

If the dominant phase separation mechanism is diffusion of particles down the gradient of chemical potential, then a treatment with the Cahn-Hilliard equation [75] is appropriate. If it is assumed that there is a potential energy $V(\phi)$ associated with the order parameter ϕ, then the evolution of the order parameter for particles of mobility M and diffusivity κ is given by:

$$\frac{\partial\phi}{\partial t} = -M\nabla^2\left[-\kappa\nabla^2\phi + \frac{\partial V}{\partial\phi}\right] \tag{2.1}$$

Following the treatment given in, for example, Gunton *et al* [76], at very early times, the order parameter may be treated as a small perturbation around its initial value of zero. Taylor-expanding the derivative of potential energy to second order then gives

$$\frac{\partial\phi}{\partial t} = -M\nabla^2\left(-\kappa\nabla^2 + \left.\frac{\partial^2 V}{\partial\phi^2}\right|_{\phi_0}\right)\phi. \tag{2.2}$$

Taking the Fourier transform:

$$\frac{\partial}{\partial t}\tilde{\phi}(k,t) = -Mk^2\left(-\kappa k^2 + \left.\frac{\partial^2 V}{\partial \phi^2}\right|_{\phi_0}\right)\tilde{\phi}(k.t) = \omega(k)\tilde{\phi}(k.t). \tag{2.3}$$

Hence, the Fourier-transformed order parameter has the form $\tilde{\phi}(k.t) = \tilde{\phi}_0(k)e^{\omega(k)t}$. The structural properties of isotropic phase-separating systems are often described by the circular or spherical average of the structure factor, defined as

$$S(\mathbf{k},t) = \left\langle \frac{1}{V}\left|\int \left(\phi(\mathbf{r}) - \bar{\phi}\right) e^{i\mathbf{k}\cdot\mathbf{r}}\right|^2 \right\rangle \tag{2.4}$$

This description is particularly useful because it may be compared directly with X-ray measurements. For phase separation in this régime, the structure factor takes the form

$$S(k,t) = S_0(k)e^{2\omega(k)t} \tag{2.5}$$

From equation 2.3, it can be seen that $\omega(k)$ is positive for wave-vectors smaller than a critical value k_c. For such wave-vectors, the structure factor retains the same shape, but grows exponentially in magnitude with time; the growth rate is a function $\omega(k)$ of wavelength. If there is a large peak in the structure factor, its position will stay the same while it grows, and there will be no change in the dominant length scale of the phase-separating system.

Once domains have formed containing a majority of one particular component, domain growth may proceed through the Lifshitz-Slyozov ripening mechanism, where droplets of the minority phase form and ripen through an evaporation-condensation mechanism, giving rise to a $t^{1/3}$ growth law [72].

Evidence for Lifshitz-Slyozov growth has been seen in Langevin models without hydrodynamics [77, 78], dissipative particle dynamics [79], Monte Carlo renormalization-group studies [80], and lattice Boltzmann studies [81, 82].

2.1.2 Viscous hydrodynamic growth

Once sharp interfaces have formed and hydrodynamics has become important, one of several growth mechanisms may come into play.

In the viscous hydrodynamic régime, the viscous term in the Navier-Stokes equation dominates over the inertial term: $\nu\nabla^2\mathbf{U} \sim 1/\rho\nabla p$, for pressure p. Assuming that dynamical scaling holds and taking $p \sim \sigma/L$ suggests that $L \sim T$. This linear growth law was predicted by Siggia [83]; however, San Miguel *et al* [84] showed that Siggia's growth mechanism would only occur in three dimensions, and not in two. Instead, they proposed a $t^{1/2}$ growth law based around interface diffusion.

For two-dimensional systems, early-time $t^{1/2}$ growth has been observed in MD simulations [85, 86],DPD [87, 39], lattice-gas automata [88], and a lattice Boltzmann model [89]. Other lattice Boltzmann models have produced a $t^{1/3}$ growth law [82, 90], which has also been observed in models which did not conserve momentum [80, 91].

2.1.3 Inertial hydrodynamic growth

If the Reynolds number of the system becomes sufficiently high, the inertial term in the Navier-Stokes equation dominates the viscous term; a brief dimensional analysis as above then suggests a $t^{2/3}$ growth law, as predicted by Furukawa [73] for times

much larger than T_0. Grant and Elder [74] suggested that for very high Reynolds numbers, turbulent remixing would slow the domain coarsening process to a $t^{1/2}$ law, but this has not yet been observed, and it has been pointed out [87] that their assumptions are debatable.

A $t^{2/3}$ late-time growth law in two dimensions has been seen in several numerical simulations, such as Ginzburg-Landau models [91], dissipative particle dynamics [79], or lattice Boltzmann models [90, 89, 92, 93].

2.2 Previous LBE studies

2.2.1 Rothman-Keller models

Grunau *et al* [89] examined the separation of two immiscible phases in porous media with wetting walls. During the early stages of separation, when the phases had not completely separated, the domain growth was found to be independent of the wall characteristics; an early-time growth exponent of $\frac{1}{2}$ was found, followed by a late-time exponent of $\frac{2}{3}$. Chen and Lookman[92] examined separation of three or four immiscible components in 2D, using a model which introduced stochastic fluctuations to mimic the effect of finite temperature on the interface. They found a domain growth exponent of 0.4 ± 0.03; for three dimensions, they found an exponent of 0.96 ± 0.05 in agreement with the predictions of Siggia.[83] Curiously, the introduction of fluctuations was found to have no effect on the domain growth.

Rybka *et al* [82] used a Rothman-Keller model to find an early-time growth exponent of 0.33 ± 0.06, and a late-time exponent of 0.66 ± 0.06 for correlated quenches.

2.2.2 Free-energy model studies

Osborn *et al* [90] examined 2D spinodal decomposition in binary fluids with a free-energy model, and found early time exponents of $\frac{1}{2}$ for early times, and $\frac{2}{3}$ for late times. Cates *et al* [94, 79] examined spinodal decomposition in 3D, finding an early-time exponent of 1, and a late-time exponent of $\frac{2}{3}$.

2.2.3 Shan-Chen models

Although phase separation has been observed in Shan-Chen models[57], there have been no quantitative examinations of the domain growth. The 2D Shan-Chen LBE code was used to probe this area, and results were obtained which agreed with existing theories for the early-time diffusive régime and late-time inertial hydrodynamic régime. However, no observations of the viscous hydrodynamic régime have been made; it has been suggested that this is because the crossover time at which inertial effects begin to dominate over diffusive effects occurs before the crossover between diffusive and viscous régimes.

2.3 Spinodal decomposition study

Spinodal decomposition simulations were performed on a 256×256 lattice. The simulations were initialized to contain a number density of $0.5 + \delta$ red particles

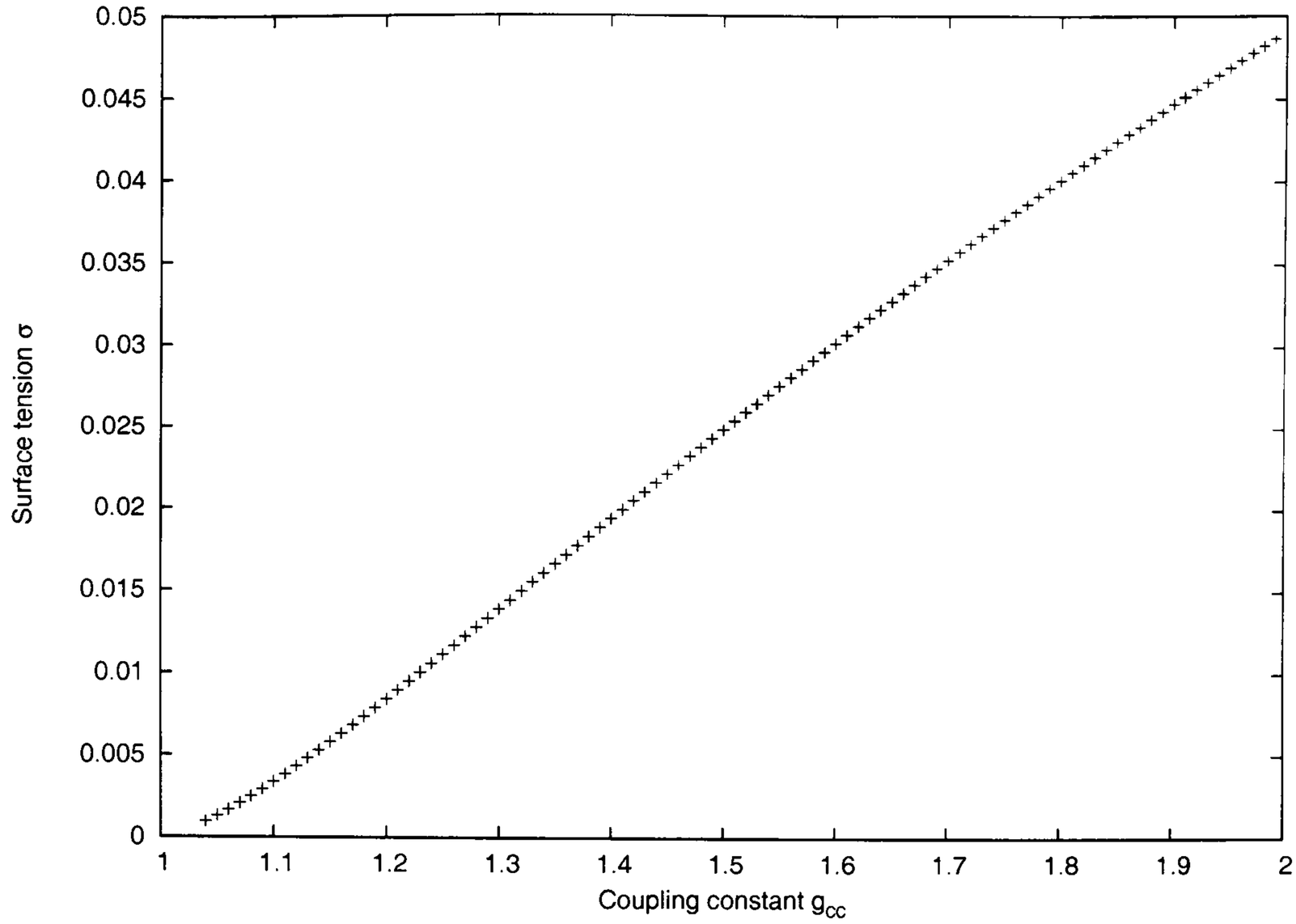


Figure 2.1: Surface tension in lattice units versus coupling constant g_{cc}, for $\rho = 1.0$ and $\tau = 1.0$; error bars are smaller than the point markers. Surface tension below $g_{cc} = 1$ is close to or equal to zero.

and $0.5 - \delta$ blue particles at each site, where δ is chosen randomly from a uniform distribution in the range $-0.025 \leq \delta \leq 0.025$ at each lattice site, to provide a small initial perturbation in the order parameter. Since there are no thermal fluctuations in this lattice Boltzmann algorithm, the system would sit forever in a metastable state if the order parameter and density were exactly uniform across the system.

A conventional measurement of the typical domain size $R(t)$ in simulations of phase separation is the inverse first moment of the circularly averaged structure factor (equation (2.4)) :

$$R(t) = 2\pi \left(\sum_k S(k) \right) / \left(\sum_k kS(k) \right) \tag{2.6}$$

In régimes where only one length scale exists, $R(t)$ should scale with the same exponent as any other measure of domain size.

2.3.1 Surface tension

In order to conduct investigations of spinodal decomposition, the surface tension σ must be determined. A convenient method for doing this is to evaluate the integral of the difference between the components of the pressure tensor perpendicular and parallel to the interface:

$$\sigma = \int_{-\infty}^{\infty} \left[\Pi_{\perp} - \Pi_{\parallel} \right] \, \mathrm{d}z = \int_{-\infty}^{\infty} \left[\Pi_{xx} - \Pi_{yy} \right] \, \mathrm{d}x \tag{2.7}$$

All simulations described in this chapter used relaxation times of $\tau^r = \tau^b = 1.0$, and a mean density of 1.0. For each value of the coupling constant $g_{cc} = g_{rb} =$

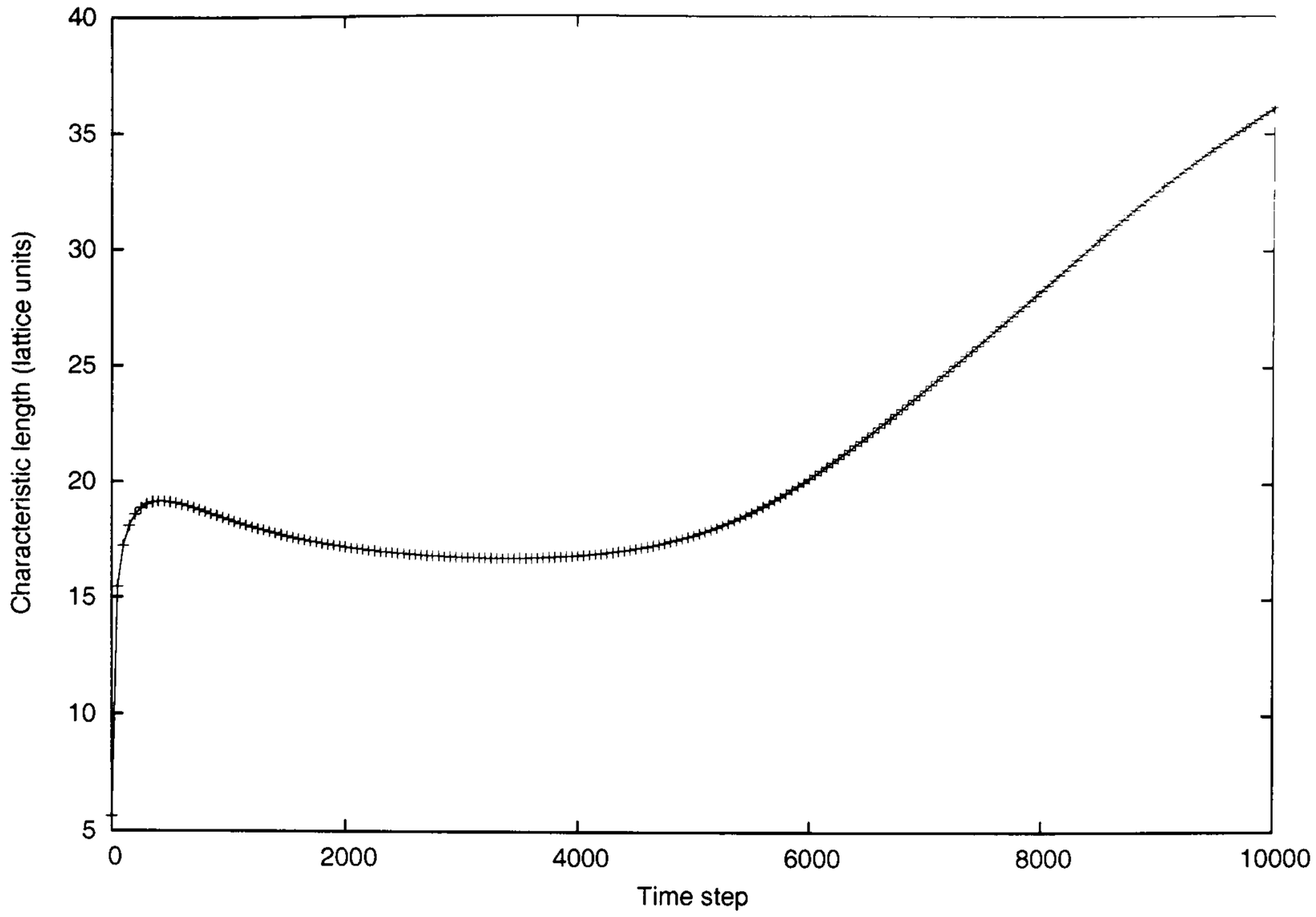


Figure 2.2: Characteristic length $R(t)$ against simulation time step, both in lattice units, for $g = 1.08$, $\rho = 1.0$,$\tau = 1.0$.

g_{br} used in a phase separation simulation, the corresponding surface tension was determined by initialising a simulation on a 128×32 grid, initially split into two 64×32 regions containing one single component, and allowing the interface to relax. The surface tension was then calculated once fluctuations had died down, usually after around 40000 time steps.

2.3.2 Interface formation

A simulation was performed with the coupling constant $g = 1.08$ giving a surface tension $\sigma = 0.0025$. For the period approximately between time steps 1000 and 5000, little change in the typical domain size $R(t)$ was observed, as can be seen in Figure 2.2. However, when the structure factor $S(k)$ is plotted as a function of wave-vector, the exponential growth predicted by a Cahn-Hilliard treatment can be clearly seen, as in Figure 2.3.

The wavelength-dependent growth rate $\omega(k)$ was determined from the results by fitting a straight line through a graph of $\log S(k,t)$ against t for each wave-vector k. The form of $\omega(k)$ obtained is close to the quartic functional form suggested in equation (2.3), as can be seen in Figure 2.4.

In Figure 2.5, it can be seen that during this stage of phase separation, the size and positions of the domains remain roughly the same,while the interfaces between each domain gradually sharpen as separation proceeds. Once sharp interfaces have formed, the domains then begin to coarsen through other mechanisms: the exponential growth in structure factor stops, and the structure factor peak moves towards longer wavelengths.

In addition to the previously-detailed analytical treatment, early-time exponential growth of the structure factor has been simulated in dissipative particle dynamics studies[40]: these results show that it is now also possible to examine the

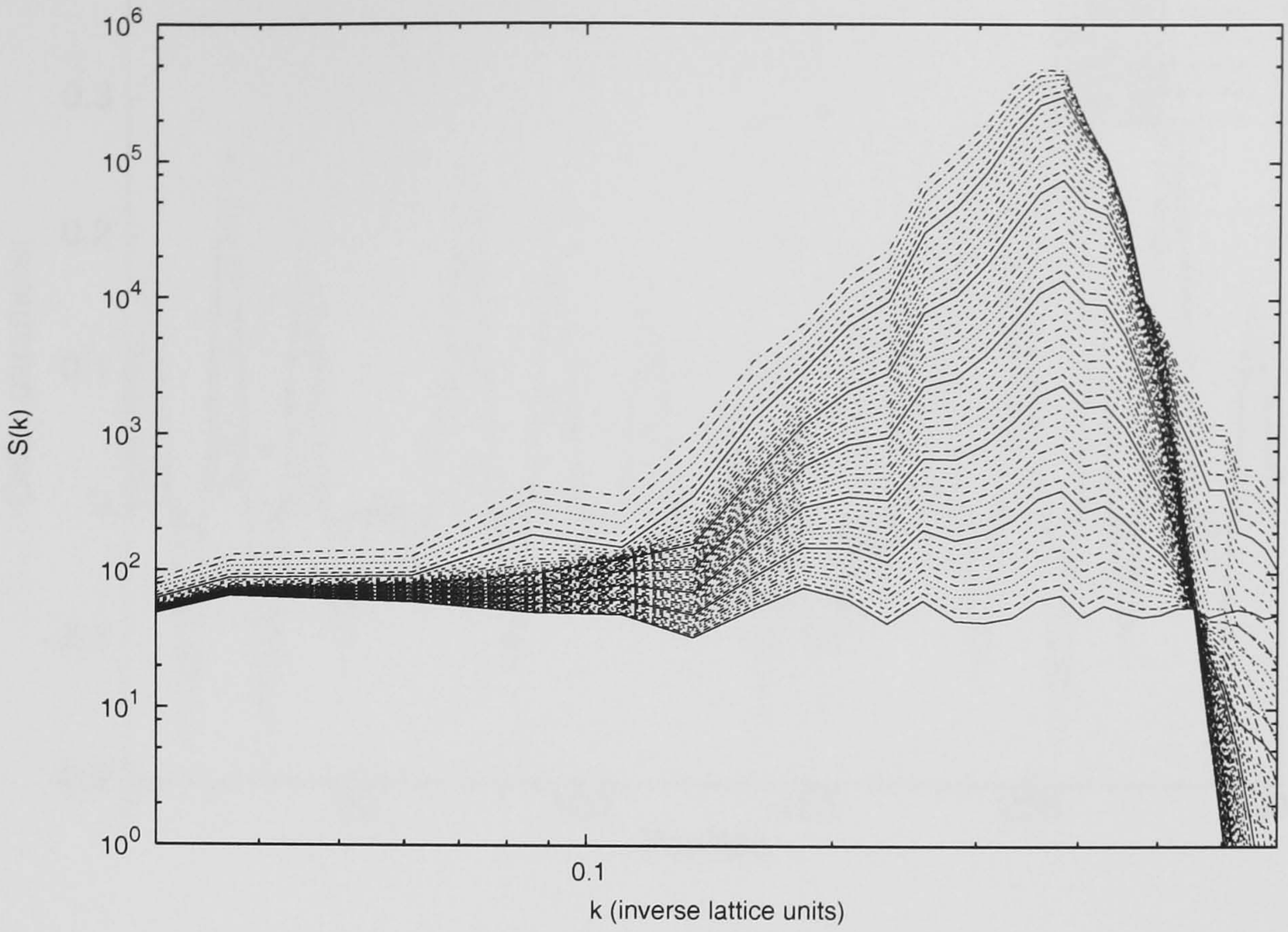


Figure 2.3: The structure factor $S(k)$ plotted against wave-vector k for time step 0 (lowest curve), every 100 time steps until time step 5000 (highest curve). Since the vertical axis is logarithmic, the even spacing of the curves demonstrates the exponential growth present at early stages of phase separation.

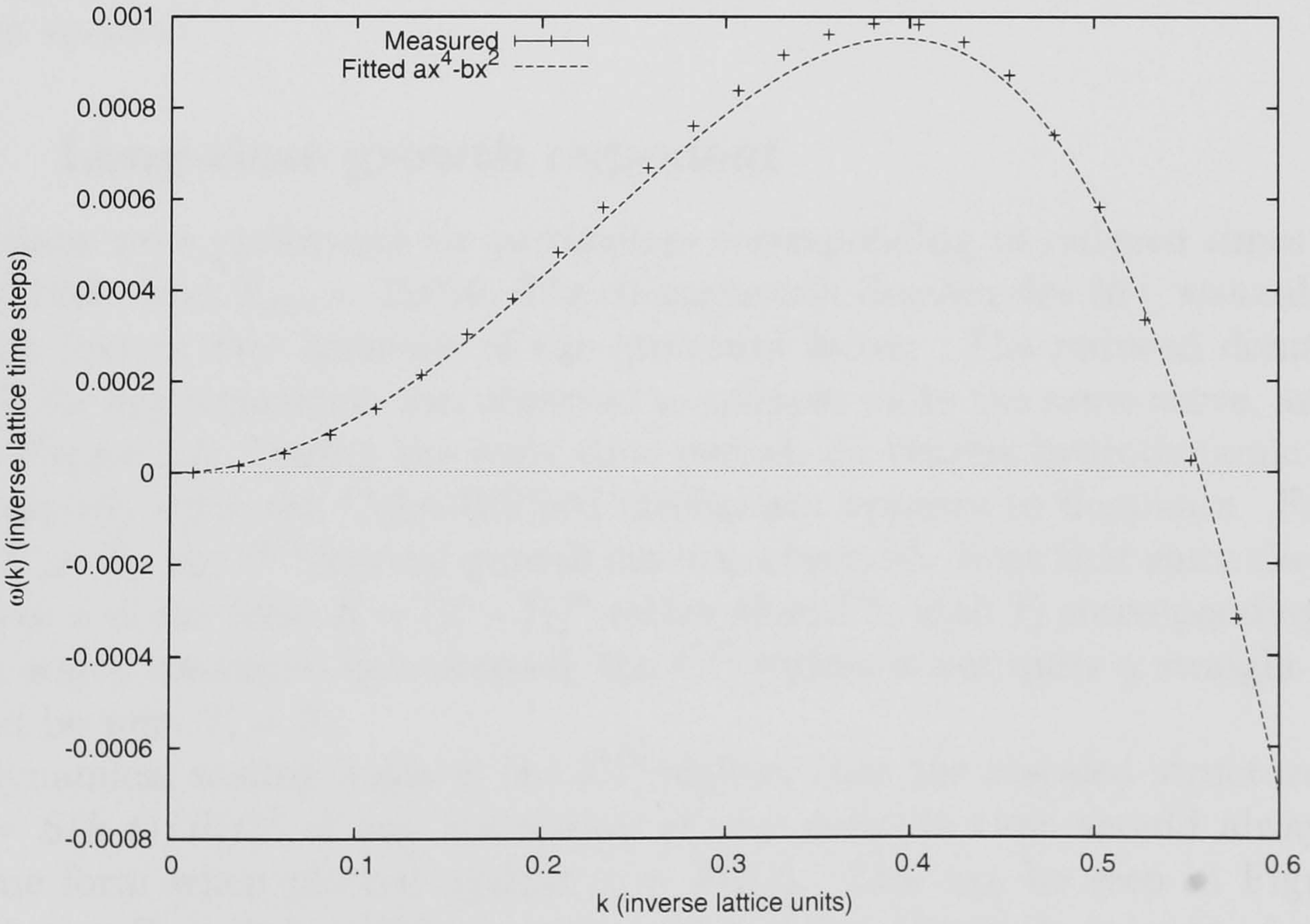
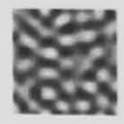


Figure 2.4: Growth rate $\omega(k)$ against wave-vector k, compared with the analytical Cahn-Hilliard form.



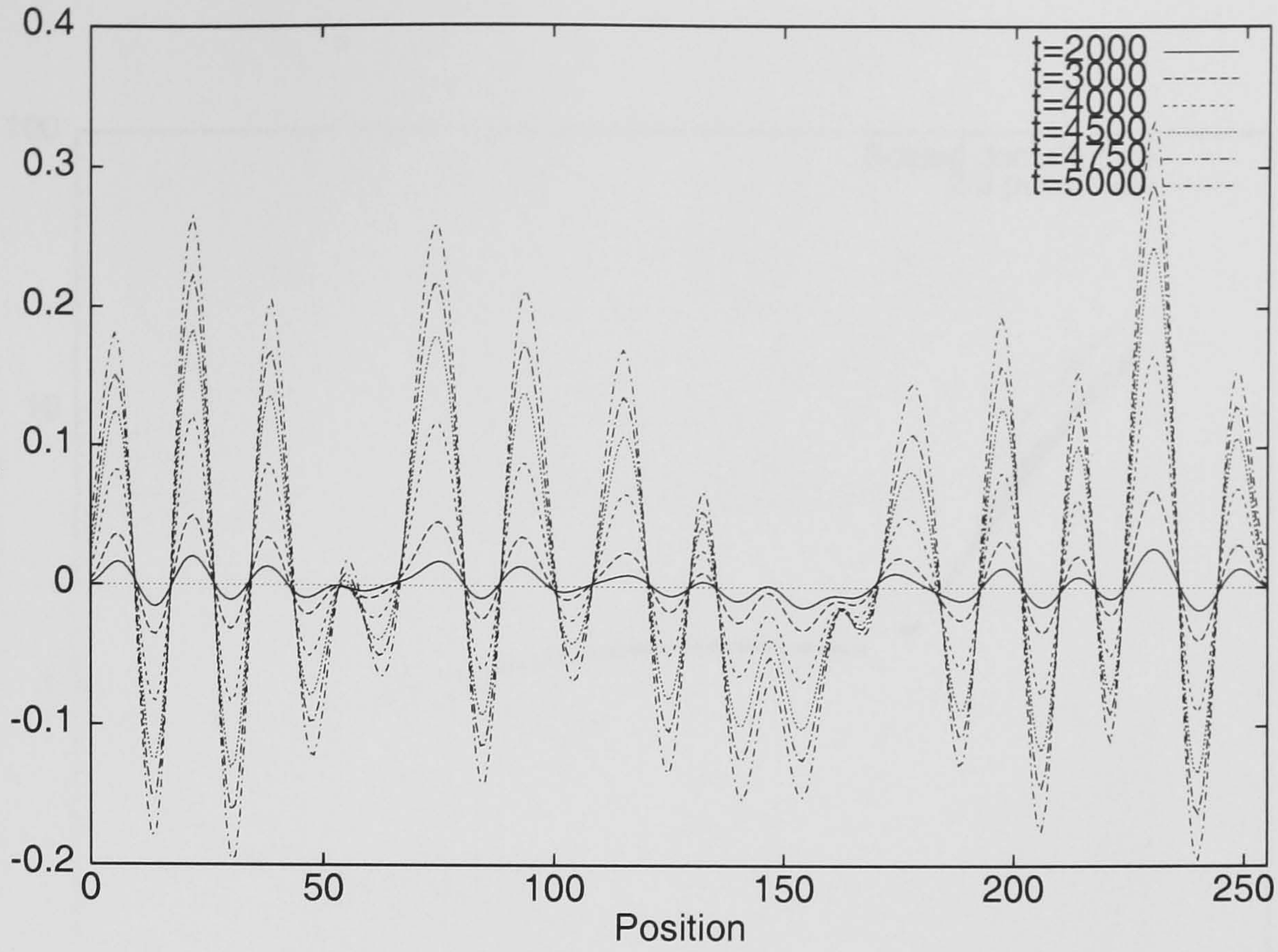


Figure 2.5: A one-dimensional cross-section of the lattice, showing the order parameter in lattice units against position in lattice units, for several different time steps during the interface formation stage. Note that the width of domains remains constant, while the depth increases.

régime with lattice Boltzmann models. It is perhaps worth noting that the results of a free-energy Cahn-Hilliard model are reproduced here by a model which does not employ an explicit free-energy functional in its implementation of interactions between species.

2.3.3 Long-time growth exponent

Simulations were performed for parameters corresponding to reduced times $0.01 \leq T/T_0 \leq 1000$, with $T_{\max} = 10000$. The characteristic domain size $R(t)$ was calculated from the inverse first moment of the structure factor. The reduced domain size $R(t)/R_0$ for any simulation was observed to collapse on to the same curve, as can be seen in Figure 2.6. During the early-time period, no viscous hydrodynamic growth was observed, since the Cahn-Hilliard mechanism appears to dominate. For large times, $T \gg T_0$, the $t^{2/3}$ inertial growth law was observed. Note that since the power-law curve is of the form $R = (T - T_i)^n$ rather than T^n, with T_i corresponding to the time at which interfaces have formed, the $t^{2/3}$ régime is not quite a straight line, as it would be were $T_i = 0$.

If dynamical scaling holds in the $t^{2/3}$ régime, then the rescaled structure factor $F(x) = S(k,t)/R(t)^2$ of any simulation at any point in time should always have the same form when plotted against $x = kR(t)$. This can be seen in Figure 2.7. According to Porod's law [80], a system with a sufficiently large amount of interface should produce a structure factor of the form $F(x) \propto x^{-3}$ for sufficiently large x in two dimensions; this curve is plotted for comparison. However, for very large wavevectors, Porod's law is not expected to be obeyed, since the corresponding length scale is similar to that of the interface width.

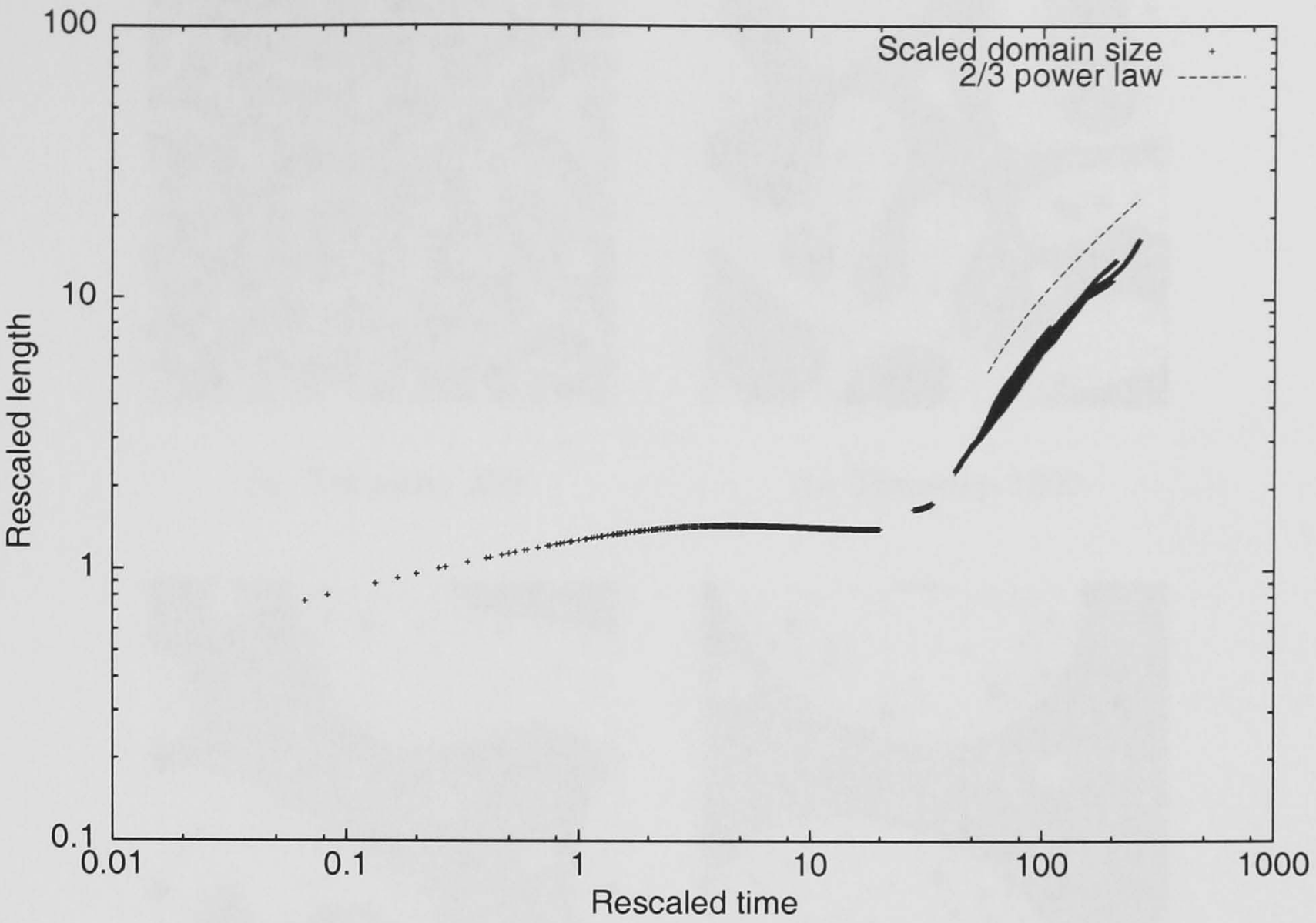


Figure 2.6: Reduced domain size plotted against reduced time.

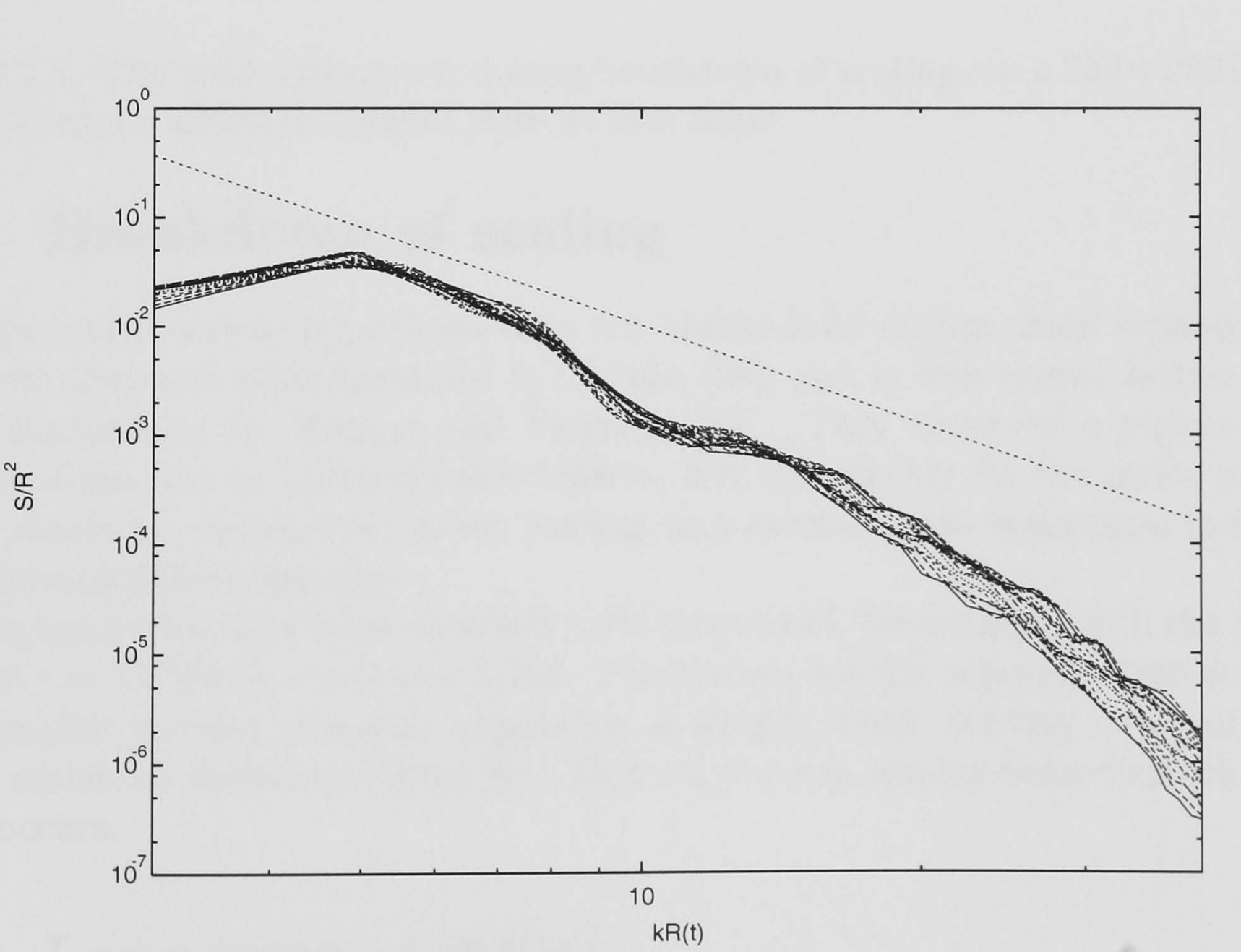
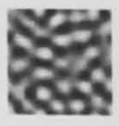
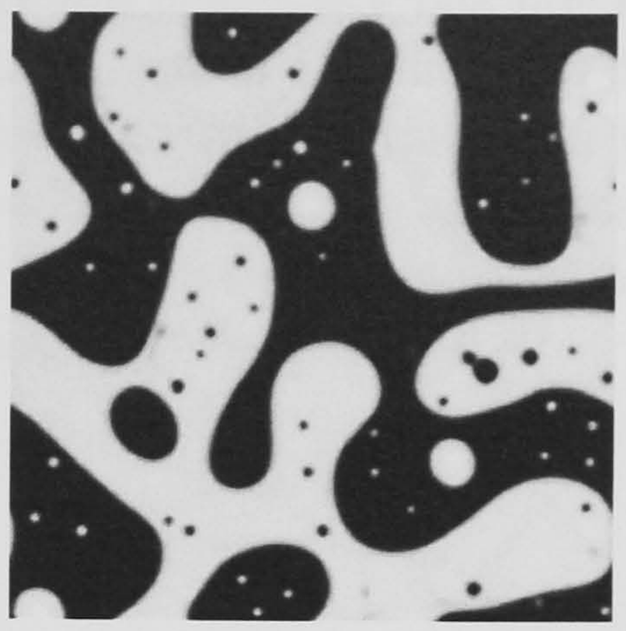


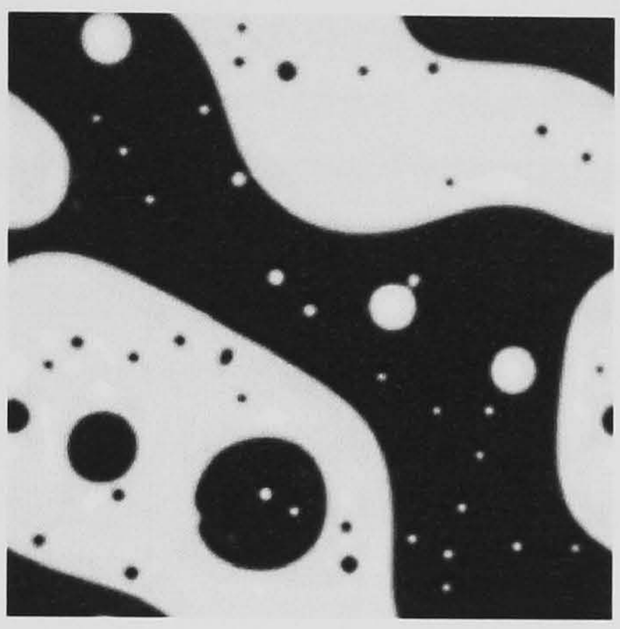
Figure 2.7: Rescaled structure factor in the scaling régime, with the dotted line representing an x^{-3} curve for comparison with Porod's law.



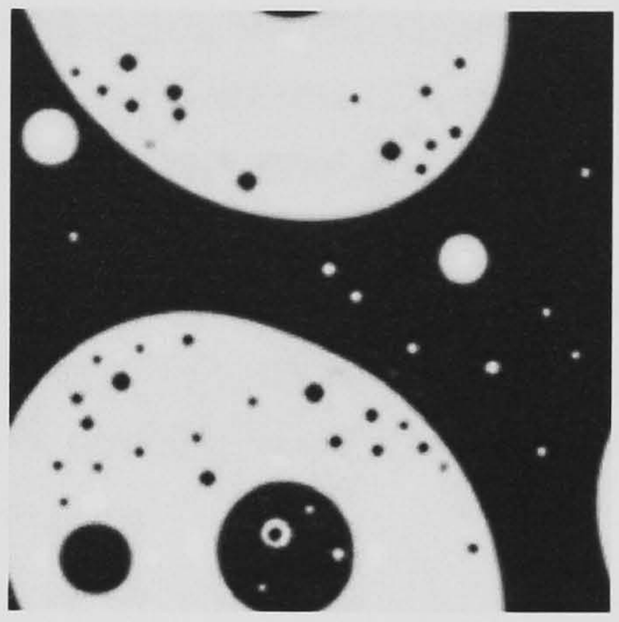
(a) Timestep 250



(b) Timestep 1000



(c) Timestep 2500



(d) Timestep 10000

Figure 2.8: The order parameter during breakdown of scaling, on a 256×256 lattice. Note the many different droplet sizes at late times.

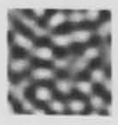
2.4 Breakdown of scaling

The dynamical scaling hypothesis does not always hold during phase separation, as has been observed experimentally by Tanaka [95], and in free-energy lattice Boltzmann simulations by Wagner and Yeomans [81]. They observed a régime where domains form due to hydrodynamic effects, but too quickly for the order parameter to obtain its equilibrium value, leading to a second phase separation inside the initially-established domains.

The same effect has been observed with this model, for example with the parameter set $\tau = 1.8765$, $\rho = 1.0$, $g = 3.266$. The system quickly separates into domains, with smaller circular domains appearing at longer times, leading to a variety of length scales, as shown in Figure 2.8. There is no clear scaling behaviour when this effect occurs.

2.5 Long-term stability

Kendon *et al* observed instabilities with a free-energy lattice-Boltzmann model [96], and suggested that any simulation using the algorithm would eventually become unstable. The Shan-Chen model used here has certainly shown instabilities, for example for systems with very high surface tensions which lead to large particle



velocities and numerical overflow. *Unconditional* instability has not been observed: systems which became unstable usually did so within the first few thousand time steps. As a further check on stability, a phase-separation simulation was performed on a 256×256 lattice for 2×10^6 time steps, without showing any signs of instability. Between time step 1.95×10^6 and time step 2×10^6, the value in lattice units of the order parameter at any lattice site differed at most by 4.8×10^{-12}.

2.6 Conclusions

When used to examine spinodal decomposition, a two-dimensional lattice Boltzmann model showed agreement with the Cahn-Hilliard theory during interface formation at very early times. Dynamical scaling was seen for the late-time inertial stage of phase separation, with the typical domain size scaling as the time raised to the power of 2/3. However, under certain circumstances, breakdown of dynamical scaling was also seen, as has been observed experimentally and in another LBE model. Viscous-régime dynamical scaling was not observed.

For certain parameter sets, the model was found to remain numerically stable for very many time steps, in contrast to a free-energy model which apparently does not display such behaviour [96]; in our model, instabilities only appear to set in for extreme values of the simulation parameters.

Chapter 3

Hele-Shaw flow

This chapter begins with a description of the Hele-Shaw cell, a piece of experimental apparatus in which a wide variety of fluid dynamical effects can be observed with ease. A modification to existing lattice Boltzmann models of flow inside a Hele-Shaw cell is presented, which permits greater accuracy and numerical stability of the model. Several simulations are described, which demonstrate that this is the case. Finally, it is shown that this model can simulate the "viscous fingering" hydrodynamic instability.

3.1 The Hele-Shaw cell

A Hele-Shaw cell typically consists of a thin layer of fluid trapped between two stationary plates, as shown in figure 3.1. The plates are separated by spacers but otherwise clamped together, and usually made of glass or clear plastic so that the fluid can be observed easily. Fluid may be driven through the cell by gravity, or by forcing fluid at the boundaries of the cell. Details of the construction of such a cell are given by Walker[97] or Saffman and Taylor[98]. Hele-Shaw[99] originally developed the cell for the purpose of producing laminar flow and visualizing flow patterns; however, as will be shown below, it turned out to be useful in other ways.

As noted by, for example, Flekkøy [100], if the plates of a Hele-Shaw cell are sufficiently close together, then it is possible to write the equations for the two-dimensional flow in closed form. Choosing a coordinate system with the x and y axes in the plane of the cell, and the z axis perpendicular to it, the three-dimensional Navier Stokes equation for the velocity $\mathbf{v}_{3D}$ is

$$\dot{\mathbf{v}}_{3D} + (\mathbf{v}_{3D} \cdot \nabla)\mathbf{v}_{3D} + \frac{1}{\rho}\nabla p = \nu \nabla^2 \mathbf{v}_{3D}. \tag{3.1}$$

If the plates are brought closer and closer together, then variation in the z

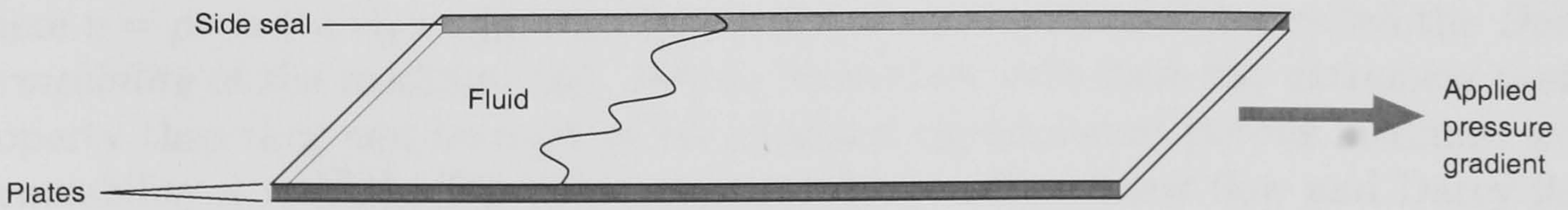
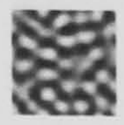


Figure 3.1: A Hele-Shaw cell. The pressure gradient can be applied by forcing fluid in from a raised feeder vessel, or by holding the cell vertically.



direction will be suppressed to the extent that one can assume that the 3D flow velocity is parallel along any line in the z direction. The velocity field then takes the form[101]

$$\begin{aligned} v_x(x,y,z) &= v_x(x,y)f(z) \\ v_y(x,y,z) &= v_y(x,y)f(z) \\ v_z(x,y,z) &= 0. \end{aligned}$$

Substituting into (3.1) yields the 2D flow equation

$$\dot{\mathbf{v}} + (\mathbf{v}\cdot\nabla)\mathbf{v} + \frac{1}{\rho}\nabla p = \nu\nabla^2\mathbf{v} + \nu\frac{\partial^2 f}{\partial z^2} \tag{3.2}$$

where

$$\mathbf{v}(x,y) = (v_x(x,y), v_y(x,y))\,. \tag{3.3}$$

Away from the edge of the cell, the velocity can be expected to take a Poiseuille profile along the z axis; applying no-slip boundary conditions at the plate edges gives

$$f(z) = 1 - \frac{4z^2}{h^2} \tag{3.4}$$

where h is the plate separation distance. This gives the closed-form description of the Hele-Shaw cell flow:

$$\dot{\mathbf{v}} + (\mathbf{v}\cdot\nabla)\mathbf{v} + \frac{1}{\rho}\nabla p = \nu\nabla^2\mathbf{v} - \frac{8\nu}{h^2}\mathbf{v} \tag{3.5}$$

This is an ordinary two-dimensional Navier-Stokes equation, with an additional term opposing and proportional to the velocity, representing the drag force exerted by the plates. It should be noted that the symbol $\mathbf{v}$ in this equation represents the velocity in the central plane of the channel; some practitioners prefer to work in terms of the averaged flow velocity $(2/3)\mathbf{v}$.
At low Reynolds number, the equilibrium flow takes the form

$$\mathbf{v} = \frac{-h^2}{8\rho\nu}\nabla p. \tag{3.6}$$

This is the famous Darcy equation for flow through a porous medium, more often stated in the form

$$\mathbf{v} = -\frac{\kappa}{\eta}\nabla p \tag{3.7}$$

where $\eta = \rho\nu$ is the dynamic viscosity, and κ is an empirical factor called the *Darcy permeability* of the medium[102]. Hence, Hele-Shaw cells have the extremely useful property that they can be used as an idealized experimental porous medium, with permeability $\kappa = h^2/8$. The close analogy between Hele-Shaw flow and Darcy flow means that a simple two-dimensional system can be used to replicate effects seen in flow through a complicated porous medium. which would be more complicated to examine experimentally, and more expensive to simulate.

3.2 Lattice Boltzmann model for Hele-Shaw flow

A number of lattice Boltzmann simulations of Hele-Shaw flow have been performed. Gondret *et al* [103] performed 3D simulations, verifying that the flow profile in the z direction was close to parabolic apart from the region very close to the side walls. Flekkøy *et al* [104, 100] performed two-dimensional lattice BGK calculations, in which an extra force term $\mathbf{F} = -(8\nu/h^2)\mathbf{v}$ was added in order to simulate the drag from the cell walls, and a very similar approach was taken by Grosfils *et al* [105, 106].

Another useful effect of the Hele-Shaw approach to flow in porous media is the fact that, in the equations of Darcy flow, the fluid viscosity never appears on its own, but only in the ratio ν/κ. For reasons of accuracy and numerical stability[107, 108], it can be difficult to simulate the flow of fluids with very high viscosities in LBGK; however, due to this convenient property of the Darcy equation, a similar effect can be obtained by giving a fluid a very low permeability. In multicomponent fluids, each component may be assigned a different effective cell width h, and correspondingly different permeability, resulting in behaviour very similar to that of fluids with differing viscosities in a real Hele-Shaw cell.

The two-dimensional lattice BGK code used in chapter 2 was modified to incorporate the Hele-Shaw drag term in a similar manner to that used by Flekkøy or Grosfils, so that it could be used to simulate multicomponent Hele-Shaw flow. However, in the course of this modification, it was discovered that, under certain circumstances, it is not quite sufficient to insert the drag term directly. As will be demonstrated below, this approach does not give the correct dynamical behaviour, and can even lead to numerical stability problems.

Recall that the Shan-Chen model contains an explicit force term $\mathbf{F}$, normally used to implement a body force to represent gravity, or to implement interacting fluids. Before the equilibrium distribution is calculated, the velocity of the fluid at a lattice site undergoes the transformation $\mathbf{v} \rightarrow \mathbf{v} + (1/\rho)\mathbf{F}\delta t$, where δt, the duration of a timestep, is normally set to unity in LBGK. This can be thought of as the local force $\mathbf{F}$ accelerating particles at the site, before they collide. Since gravitational acceleration is constant, and since interaction forces depend on local composition gradients, which are unchanged over the forcing step, the force is effectively a constant over the duration of its application.

However, the Hele-Shaw drag force is velocity-dependent, so that if the velocity varies significantly over a single timestep, then the drag force will also vary. If it is assumed that the drag force is constant over a timestep, then the resulting dynamics will be incorrect.

Suppose that an element of fluid has total momentum $\mathbf{p}_0$, and is then accelerated by both body and drag forces during the forcing step, of duration δt, to momentum $\mathbf{p}_0 + \delta\mathbf{p}$. This can be represented by the differential equation

$$\frac{d}{dt}\mathbf{p} = \mathbf{F}_0 - \frac{8\nu}{h^2}\mathbf{p} = \mathbf{F}_0 - \frac{1}{\tau_{\mathrm{HS}}}\mathbf{p}. \tag{3.8}$$

where

$$\tau_{\mathrm{HS}} \doteq \frac{h^2}{8\nu} \tag{3.9}$$

represents the characteristic time over which the fluid relaxes to the terminal velocity at which the Hele-Shaw drag balances the velocity-independent forces.

Integrating equation (3.8) over the interval from $\mathbf{p} = \mathbf{p}_0$ at time t to $\mathbf{p} = \mathbf{p}_0 + \delta\mathbf{p}$ at time $t + \delta t$ yields

$$\delta\mathbf{p} = \mathbf{p}(t + \delta t) - \mathbf{p}(t) = (\tau_{\mathrm{HS}}\mathbf{F}_0 - \mathbf{p}_0)\left(1 - e^{-\delta t/\tau_{\mathrm{HS}}}\right). \tag{3.10}$$

Expanding the exponential in a Taylor series gives

$$\delta\mathbf{p} = \mathbf{F}_0\delta t - \mathbf{p}\frac{\delta t}{\tau_{\mathrm{HS}}} + O\left(\frac{\delta t}{\tau_{\mathrm{HS}}}\right)^2 \tag{3.11}$$

so that for thicker and thicker cell widths h, the relaxation timescale τ_{HS} increases, and it is relatively safe to assume a constant drag over a timestep, so that the method of Flekkøy is recovered.

However, if the velocity field is changing on a timescale of order τ_{HS}, this assumption no longer holds: it corresponds to numerically integrating the velocity field over the forcing step using a crude forward-Euler scheme, whereas equation (3.10) uses the analytical result.

3.2.1 Multicomponent implementation

A Shan-Chen lattice BGK fluid with sound speed c_s and relaxation time τ_σ will have a viscosity of $(\tau_\sigma - 1/2)/3$. A Chapman-Enskog procedure can be used[109] to show that the viscosity of a multicomponent mixture is

$$\nu = \sum_\sigma x_\sigma c_s^2(\tau_\sigma - \frac{1}{2}) \tag{3.12}$$

where $x_\sigma = n_\sigma/n$ is the volume fraction of component σ. It should be noted that this is not necessarily how the viscosity of a mixture behaves in real fluids, and exponential mixing laws[107, 110] have also been used. This investigation used a linear mixing rule, since that corresponds to the behaviour of an unmodified Shan-Chen fluid; however, it could also be modified to use other rules. Since the viscosity of a mixture is linearly additive, the drag term for a mixture was chosen to be additive as well: this is equivalent to making the viscosity-permeability ratio ν/κ additive. With this choice, an element of fluid of momentum $\mathbf{p}$ with fraction x_σ of component σ with viscosity ν_σ and effective cell width h_σ experiences a drag force

$$\mathbf{F}_{\mathrm{HS}} = -8\mathbf{p}\sum_\sigma \frac{x_\sigma\nu_\sigma}{h_\sigma^2}. \tag{3.13}$$

This gives a multicomponent Hele-Shaw relaxation time of

$$\tau_{\mathrm{HS}} = \left(8\sum_\sigma \frac{x_\sigma\nu_\sigma}{h_\sigma^2}\right)^{-1}. \tag{3.14}$$

Over a single timestep, the total force on the fluid due to Hele-Shaw drag is then

$$\mathbf{F}_{\mathrm{drag}} = \frac{1}{\delta t}(\tau_{\mathrm{HS}}\mathbf{F}_0 - \mathbf{p}_0)\left(1 - e^{-\delta t/\tau_{\mathrm{HS}}}\right) - \mathbf{F}_0. \tag{3.15}$$

This is applied to each component in proportion to the volume fractions:

$$\mathbf{F}_{\sigma(\mathrm{drag})} = x_\sigma\mathbf{F}_{\mathrm{drag}} \tag{3.16}$$

so that the total drag force on the fluid at a lattice site remains $\mathbf{F}_{\mathrm{drag}}$.

3.3 Hele-Shaw flow simulations

Several simulations were performed to compare the simulated Hele-Shaw flow with easily-derived analytical results, and to compare the behaviour with a corrected drag term to that of uncorrected simulations.

3.3.1 Velocity decay

If the fluid in a Hele-Shaw cell is in uniform motion at velocity $\mathbf{v}_0$, then in the absence of other forces the Hele-Shaw drag will decelerate it to rest, with the evolution given by

$$\mathbf{v}(t) = \mathbf{v}_0 e^{-t/\tau_{\mathrm{HS}}}. \tag{3.17}$$

Lattice BGK simulations of this scenario were initialised to contain fluid with a local Maxwellian distribution corresponding to a velocity vector of $(0, 0.01)$ in lattice units. The fluid had red particles at a uniform density of $\rho^{\mathrm{r}} = 1/3$ and blue at a uniform density of $\rho^{\mathrm{b}} = 2/3$, giving an order parameter of $\phi = \rho^{\mathrm{r}} - \rho^{\mathrm{b}} = -1/3$, although the simulations were later repeated with order parameters of $\phi \in \{-1, -1/2, 0, 1/2, 1\}$, giving velocity fields identical to within floating-point precision. The colour-colour coupling constant g_{cc} was set to zero in all cases, so that the different components would mix but not interact or form interfaces.

The velocity decay as simulated without the corrected drag term is shown in figure 3.2, for several values of the cell width. As the cell width decreases, the Hele-Shaw relaxation time τ_{HS} decreases to the point where it is poorly resolved during a lattice Boltzmann timestep, and the deviation from the analytical solution increases correspondingly.

Figure 3.3 shows velocity decay as simulated with a corrected drag term (equation 3.10). The velocity was found to agree with the analytical solution (equation 3.17) to within floating-point error. The discrepancy between corrected and uncorrected simulations is particularly clear for the $h = 1$ case, shown in figure 3.4. For unit BGK relaxation time, $h = 1$ gives a damping timescale $\tau_{\mathrm{HS}} = 3/4$, less than an LBGK timestep: the resolution is therefore so poor than an uncorrected simulation overshoots the drag force, and the fluid rather unphysically oscillates back and forth before coming to rest.

3.3.2 Numerical stability

Consider the momentum change at a lattice site at each timestep during a simulation of drag deceleration in a Hele-Shaw cell. In the absence of other forces, using the uncorrected model of Flekkøy , $\delta\mathbf{p} = -\mathbf{p}/\tau_{\mathrm{HS}}$: if $\tau_{\mathrm{HS}} < 1$, then the momentum is numerically unstable, and grows exponentially as shown in figure 3.5, very quickly reaching completely unphysical speeds and correspondingly negative values of the distribution function. It is straightforward to show that the magnitude of momentum of the system should grow as $(1 - 1/\tau_{\mathrm{HS}})^t$: figure 3.6 confirms this. With the correction of equation (3.10) made, simulations with hitherto unstable values of τ_{HS} were observed to be stable; for example, it was possible to perform simulations at $h = 10^{-2}$, which were numerically unstable without the correction.

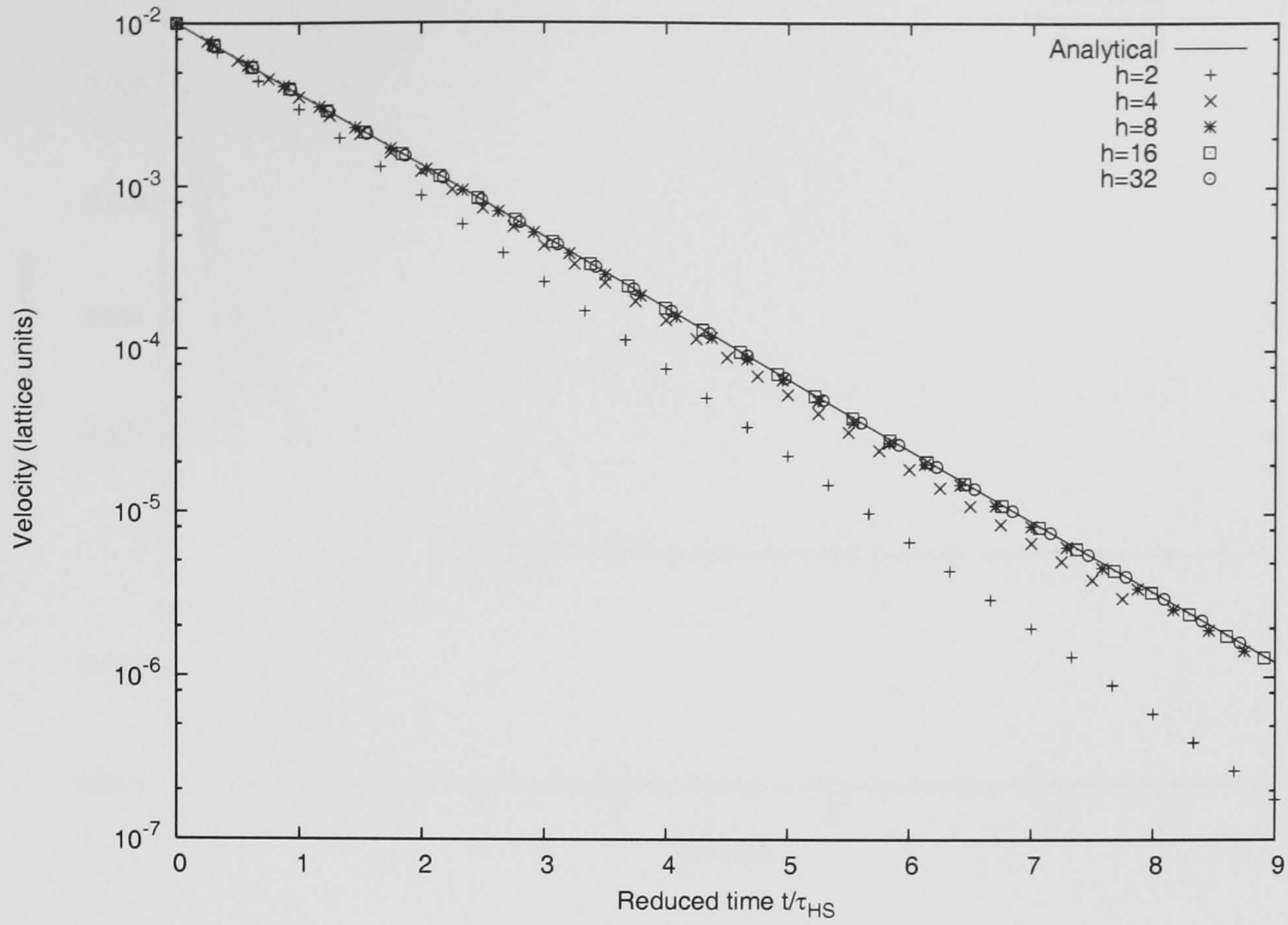


Figure 3.2: The velocity of fluid decelerating in a Hele-Shaw cell plotted against time, simulated using lattice BGK plus a simple forcing term to represent the drag force.

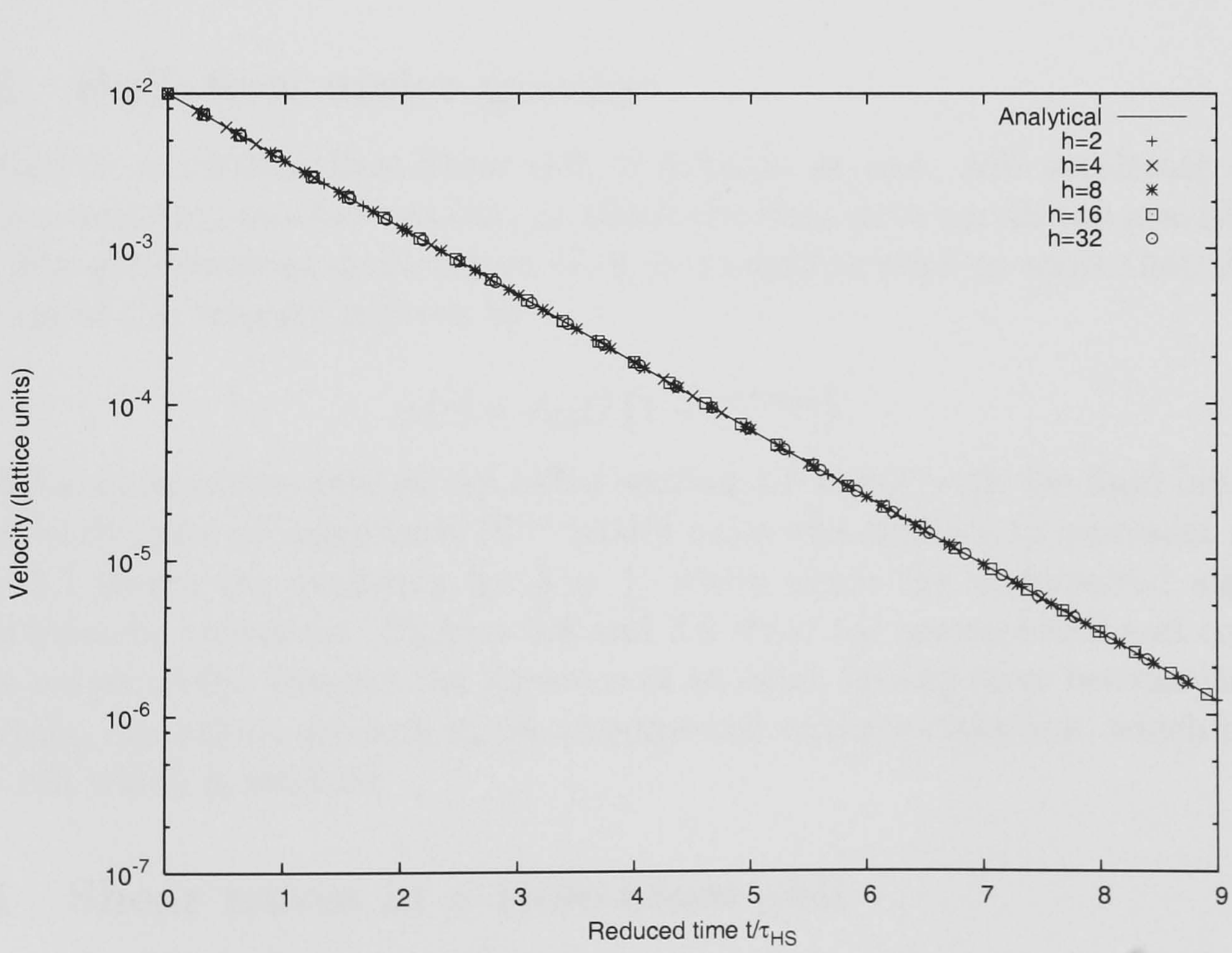


Figure 3.3: The velocity of fluid decelerating in a Hele-Shaw cell plotted against time, simulated using lattice BGK plus a drag term corrected to account for variation of the drag force during a single lattice BGK timestep.

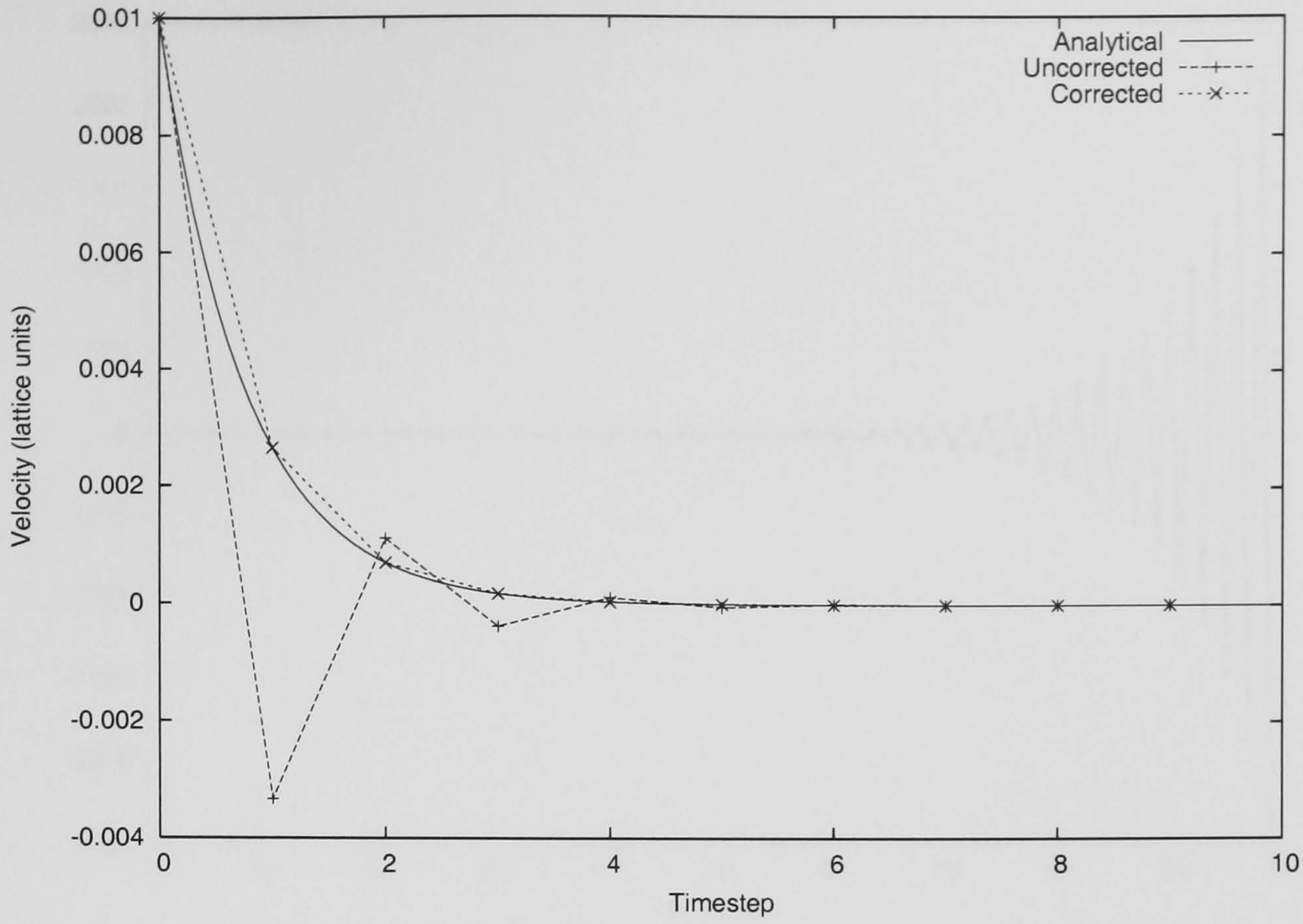


Figure 3.4: Velocity decay in a Hele-Shaw cell, showing the analytical solution, values from a simple simulation, and values from a simulation with corrected drag term. Because the Hele-Shaw relaxation time is so short in this case, the time-variation of the drag force is so poorly resolved in an uncorrected simulation that the velocity overshoots.

3.3.3 Bulk flow under gravity

The fluid in a vertical Hele-Shaw cell, if initially at rest, will accelerate until it reaches a uniform constant velocity, at which the drag force equals the gravitiational force. For gravitational acceleration G, it is straightforward to show that the time evolution of the velocity is given by

$$v(t) = \tau_{\mathrm{HS}} G \left(1 - e^{-t/\tau_{\mathrm{HS}}}\right). \tag{3.18}$$

LBGK simulations were set up as for section 3.3.1, but with the fluid initially at rest. A body force of magnitude 10^{-4} lattice units was applied, to represent gravity. Figure 3.7 shows the evolution for $h = 1$, where again the uncorrected algorithm gives a velocity overshoot. Figures 3.8 and 3.9 show the uncorrected and corrected results, respectively. Despite the presence of an extra forcing term besides the Hele-Shaw drag, deviations are seen in the uncorrected velocity evolution, which increase as the cell width is reduced.

3.3.4 Shear waves in a Hele-Shaw cell

Suppose a fluid is set up to have a sinusoidal velocity field, such as

$$\mathbf{v}(x, y) = (0, v(x)) = (0, Ae^{ikx}) \tag{3.19}$$

The 2D Hele-Shaw Navier Stokes equation reduces to the form

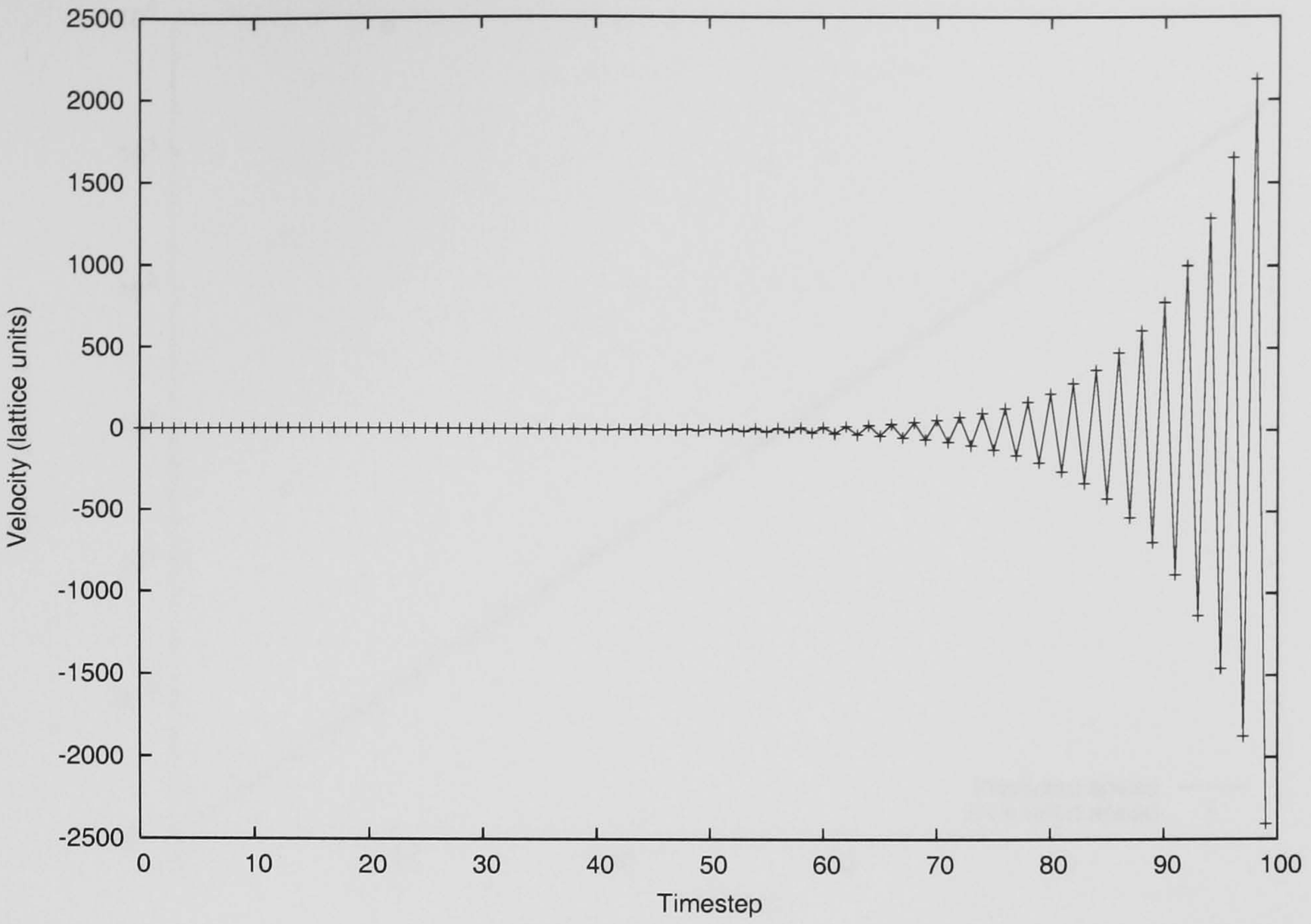


Figure 3.5: Velocity instability in an uncorrected Hele-Shaw simulation, with $h = 1$ and $\nu = 4/15$, giving $\tau_{\mathrm{HS}} = 15/32$.

$$\frac{\partial v}{\partial t} - \nu \frac{\partial^2 v}{\partial x^2} + \frac{8\nu}{h^2} v = 0. \tag{3.20}$$

This has the solution

$$v_k(x,t) = A_k e^{ikx} e^{-\nu k^2 t} e^{-t/\tau_{\mathrm{HS}}}. \tag{3.21}$$

Therefore, a shear wave will decay exponentially in time, at a rate $\omega(k) = \nu k^2$ proportional to the square of its wave-vector. Conveniently, equation (3.20) is linear, so any shear wave can be written as the sum of Fourier components A_k, each decaying according to the dispersion relation $\omega(k)$. In the absence of Hele-Shaw drag, this process is solely due to the viscous transport of momentum from the peaks to the troughs of the wave: equation (3.20) reduces to the diffusion equation. However, in a Hele-Shaw cell, there is an additional sink term, which gives rise to a wavelength-independent part of the dispersion relationship.

To calculate the dispersion relation, simulations were performed with fluid of uniform density and composition on a 128×2 grid. For each simulation, the fluid was initialised to contain a shear wave with wave-vector $k = 2n\pi/L$, with $L = 128$ and $n = 1, 2, \ldots$. The fluid was then allowed to evolve in time, and the amplitude $a(t)$ of the shear wave measured at each timestep; the simulation was terminated once this fell below a velocity of around 10^{-12} lattice units. A straight line was fitted to the graph of $\log a(t)$ against t, the gradient of the line giving the value of the decay rate ω. This was repeated for several values of the cell width h, and for the wavenumber n, and for simulations both with and without the drag correction.

Figure 3.10 shows the simulation results, in comparison to the analytical result. In all calculations, the corrected simulations produced a dispersion relation that at all points was within 2% of the analytical prediction. However, the uncorrected

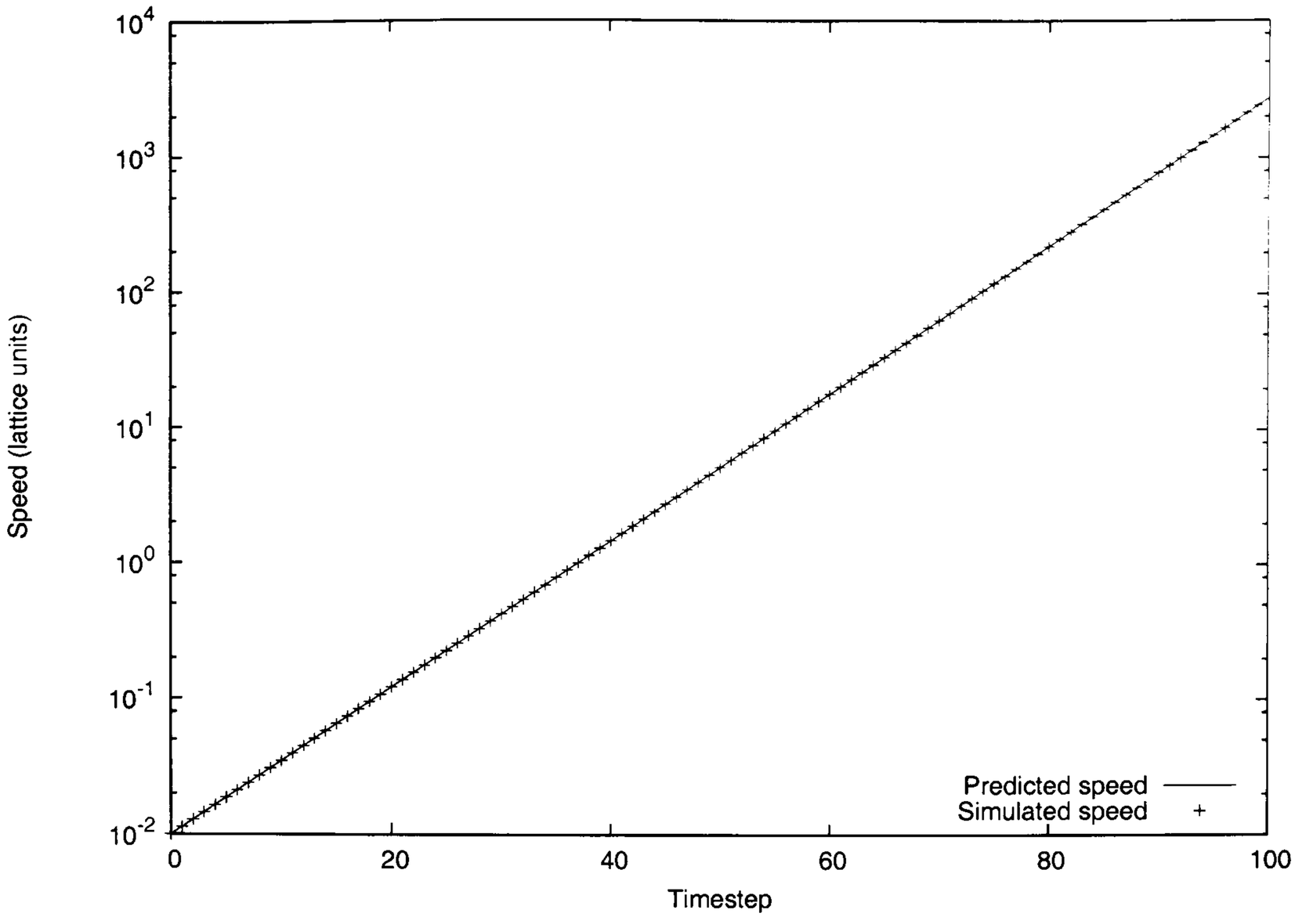


Figure 3.6: Theoretical and calculated instability growth.

simulations deviated significantly: while the quadratic wavelength dependence is reproduced, they over- or under-estimate the wavelength-independent part by a constant amount, which increases as the cell gap is narrowed.

3.4 The Saffman-Taylor instability

The modified Hele-Shaw algorithm was finally used to conduct a brief and preliminary examination of the Saffman-Taylor instability. Consider a Hele-Shaw cell filled with a fluid of high viscosity, such as honey or heavy oil. If a fluid of relatively low viscosity, such as air or water, is then forced into the cell, driving the high-viscosity fluid out, then the interface between the two fluids will usually become unstable: an initially flat interface will wrinkle and deform into many *fingers* of low-viscosity fluid which penetrate the high-viscosity fluid very quickly. In some cases, a single finger comes to dominate; in others, a *tip-splitting* effect sets in: the fingers themselves become unstable, and split into smaller fingers.

This effect is known as *viscous fingering*. The first scientific examination of the effect is often attributed to Saffman and Taylor[98], who studied air fingering into glycerine and water fingering into oil, in a Hele-Shaw cell. However, they acknowledged that the effect had long been known to mining engineers and geologists, since it can cause significant problems in the oil extraction process; in addition, Hill[111] described the instability affecting the process of sugar refinement some years before the study of Saffman and Taylor.

Homsy[112] reviewed much of the experimental work on viscous fingering, along with theoretical arguments and some of the early simulation work; further theoretical results were reviewed by Bensimon *et al* [113] and Tanveer[114]. Attempts have been made to model the instability using spectral methods[115, 116, 117], CFD[118], phase-field models[119, 120], diffusion-limited aggregation[121, 122], lattice-gas automata[123

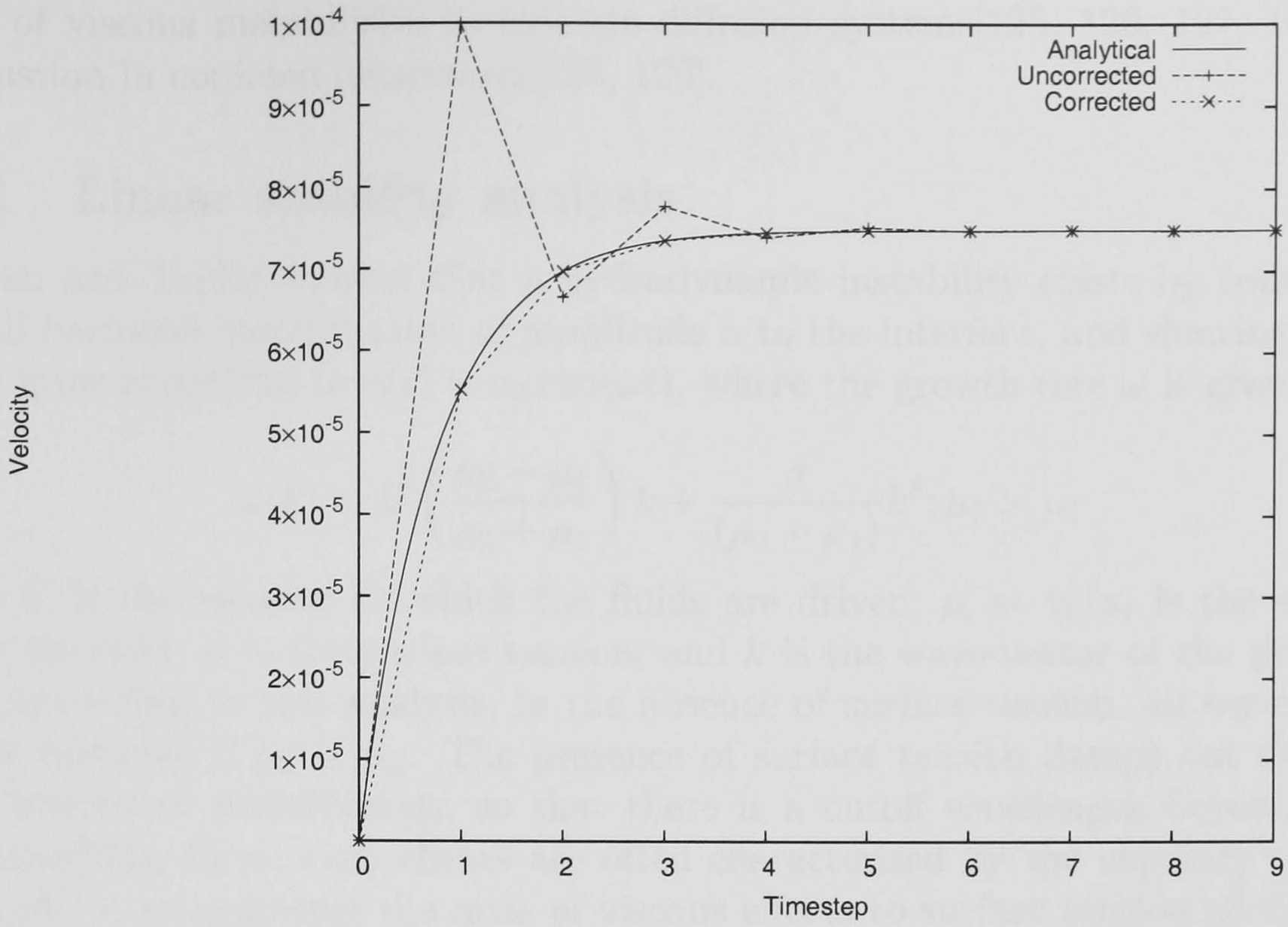


Figure 3.7: Acceleration under gravity in a Hele-Shaw cell, with analytical, simulated, and corrected simulated results shown.

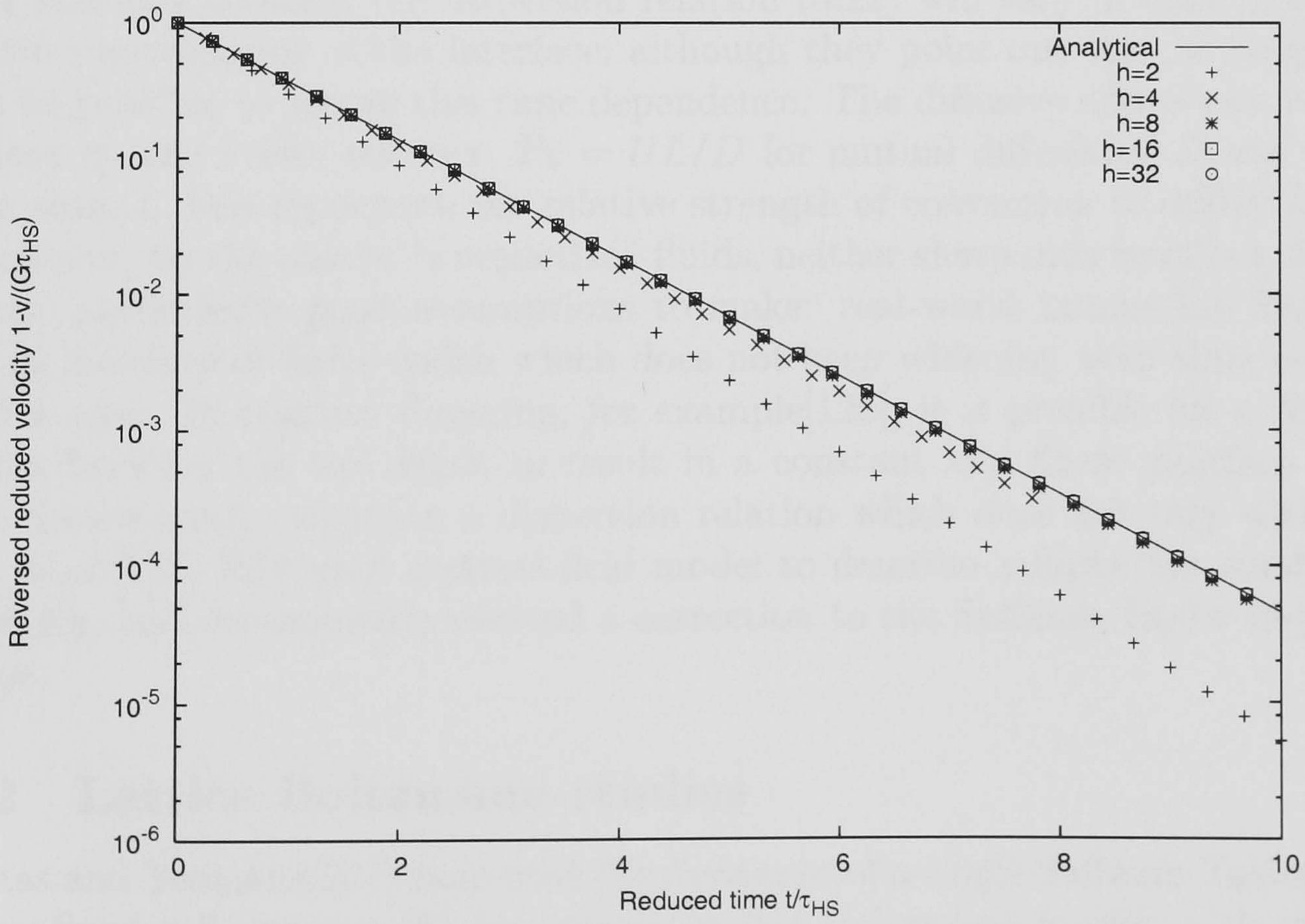


Figure 3.8: The velocity of fluid acccelerating under gravity in a Hele-Shaw cell plotted against time, simulated using lattice BGK plus a simple forcing term to represent the drag force.

and lattice Boltzmann simulations[107, 124, 108, 106, 105]. Studies have also been made of viscous instabilities in reaction-diffusion systems[125, 126, 127. 105] and combustion in confined geometries[128, 129].

3.4.1 Linear stability analysis

Saffman and Taylor showed that a hydrodynamic instability exists by considering a small harmonic perturbation of amplitude a to the interface, and showing that it would grow according to $a(t) = a_0 \exp(\omega t)$, where the growth rate ω is given by

$$\omega(k) = U\left(\frac{\mu_2 - \mu_1}{\mu_2 + \mu_1}\right)k + \frac{\sigma}{(\mu_2 + \mu_1)}k^3; \mu_2 > \mu_1 \tag{3.22}$$

where U is the velocity at which the fluids are driven, $\mu_i = \eta_i/\kappa_i$ is the effective Darcy viscosity, σ is the surface tension, and k is the wave-vector of the perturbation. According to this analysis, in the absence of surface tension, all wavelengths will be unstable if $\mu_2 > \mu_1$. The presence of surface tension damps out the more rapid interfacial disturbances, so that there is a cutoff wavelength beyond which no instabilities form; such effects are often characterised by the capillary number, $Ca = \mu U/\sigma$, representing the ratio of viscous effects to surface tension effects. The higher the capillary number, the less the instability is inhibited by surface tension.

The situation is complicated by the fact that the Saffman-Taylor viscous instability is not the only active mechanism in many of the physical systems which demonstrate fingering. For example, the above analysis assumes that there is a sharp interface between the two fluids, yet fingering can also be seen in fluids which are completely miscible[126]. Tan and Homsy[110] showed that for two fluids which diffuse into one another, the dispersion relation (3.22) will vary in time due to the diffusive smearing-out of the interface, although they point out that at long times, it can be possible to ignore this time dependence. The diffusive effects can be characterised by the Péclet number, $Pe = UL/D$ for mutual diffusivity D and typical length scale L; this represents the relative strength of convection to diffusion.

However, for the case of "immiscible" fluids, neither sharp interfaces nor diffusive ones are particularly good assumptions to make: real-world immiscible fluids will have an interface of finite width which does not keep widening with time as in the diffusive case. In reactive fingering, for example[126], it is possible for a chemical reaction between the two fluids to result in a constant and finite interface width, which consequently results in a dispersion relation which does not vary with time. Folch *et al* [119, 120] used a phase-field model to describe a finite but small interface width, and consequently derived a correction to the Saffman-Taylor dispersion relation.

3.4.2 Lattice Boltzmann studies

Langaas and Yeomans[107] examined the dynamics of a single Saffman-Taylor finger in a confined cell, using a 2D free-energy lattice Boltzmann model, with viscosity ratio up to 100, but without any Hele-Shaw drag. Rakotomalala *et al* [124] performed a similar study using a passive-scalar BGK model with viscosity ratio up to 1000. Grosfils and Boon[106, 105] extracted the early-time dispersion curve for miscible, reactive, and finite-interface-width fluids, but made no quantitative examination of

the curve. In fact, their result for zero surface tension still shows a finite cutoff wavelength, presumably due to interfacial diffusion.

3.4.3 Reproduction of viscous fingering

To verify that the adjusted-drag Hele-Shaw lattice Boltzmann code could also simulate viscous fingering, a 256×256 simulation was performed with $\tau_r = \tau_b = 1$ giving $\nu = 1/6$, and effective Hele-Shaw cell widths $h_r = 1$, $h_b = 10$ giving an effective permeability ratio of 100. The coupling constant g_{cc} was set to 1.0, chosen (see figure 2.1) to give a low mutual diffusivity $D \sim 10^{-1}$ and also a low surface tension $\sigma < 10^{-3}$. The system was initialized to contain fluid at a density $\rho = 1$, all blue at the top and all red at the bottom. The interface between the two fluids was set to white noise with a maximum amplitude of 10, and a body force of 1/750 lattice units was applied. Periodic boundary conditions were applied, to avoid confusion with effects from the side walls; this is perhaps analogous to the periodic Hele-Shaw experiments of Zhao and Maher[130].

The composition field output from this simulation is shown in figure 3.11. The initially rather rapidly varying boundary between components quickly settled down to a sharp but rough interface. By timestep 5000, most of the shorter wavelengths had been suppressed, leaving ripples which developed into clear fingers by timestep 7500. These fingers rapidly penetrated through the low-permeability fluid. Simulations performed with $h_r = h_b$ but otherwise identical parameters had interfaces which became flat within a few hundred timesteps, and remained so, with one fluid uniformly flushing the other out: this demonstrates that the instability is due to the permeability difference. The fingers appear structurally similar to those observed in experiments of early-time fingering[131, 130, 132].

While the interface between different fluid components has finite width in these simulations, for some purposes it can be useful to define a sharp interface as the curve along which $\phi = 0$. It is straightforward to find the locus of this curve by interpolation. The difference between the maximum and minimum values of the y-coordinate of this curve then gives a measure of the amplitude of the interface, sometimes referred to as the width of the "mixing zone" containing the fingers, as depicted in figure 3.12. This is plotted against time in figure 3.13, where it is clear that the interface continuously increases in amplitude due to the fingering instability. The initial drop in interfacial amplitude over the first hundred timesteps is attributed to the fact that the simulation is initialised with an unphysically sharp interfacial profile, which must relax to a finite width through diffusion. This is followed by a period of rapid growth of well-defined fingers. The interfacial amplitude is plotted for later times in figure 3.14, along with a linear growth law and a power law for comparison. The growth is somewhat faster than linear; a power law with an exponent of 1.463 ± 0.002 was fitted to the growth for $6000 < t < 20000$; Maher[131] found an exponent of around 1.6 for an experimental system with low viscosity contrast; Tryggvason and Aref[133, 134] found roughly linear growth numerically. However, the lattice Boltzmann simulation described here was performed primarily to verify the presence of the instability, and barely spans sufficiently large length or time scales for very serious comparison.

3.4.4 Dispersion relation

An approximate dispersion relation $\omega(k)$ was calculated for a range of values of the coupling constant g_{cc}. For each value of the coupling constant, several simulations were performed on a 512×128 lattice. Each such simulation was initialised to contain a sharp interface between the two fluids, set to the shape of a sine wave, with amplitude 4 lattice sites, and with wave-vector $k = 2n\pi/L$ for $L = 512$ and n an integer between 1 and 24. The simulation was otherwise performed with the same parameters as in section 3.4.3. Every 10 timesteps, the interfacial amplitude was calculated, resulting in measurements of the amplitude $a_k(t)$ of an interfacial wave of known wave-vector k as a function of time. Amplitude time-series are plotted in figures 3.15–3.19.

For $g_{cc} < 1$, the fluids are effectively miscible, to the extent that all wavelengths are rapidly suppressed by diffusion; hence, all wavelengths plotted in figure 3.15 die off exponentially, until their amplitudes become sufficiently small that there is no longer any kind of clearly-defined interfacial wave. Raising the coupling constant reduces the diffusivity and therefore reduces the rate at which the waves decay, giving slower decay rates for $g_{cc} = 0.5$ in figure 3.16. Around $g_{cc} = 1$, the components cease to be completely miscible, and certain wavelengths grow due to the Saffman-Taylor instability, as can be seen in figure 3.17. As g_{cc} is raised further, more clearly defined viscous fingering occurs: figure 3.18 shows all but the shortest wavelengths growing. As predicted by Tan and Homsy[110], the rate of growth varies in time, but settles at a constant value at later times. Some fluctuations can be seen in the mixing zone width measurements in figures 3.17–3.19, growing larger as surface tension is increased: these are attributed to capillary waves forming at the interface. In figure 3.19, the shorter wavelengths decay again, due to surface tension effects.

The amplitude time series $a_k(t)$ were observed to be generally free of early-time variations by around timestep 300 of all the simulations, as can be seen in figures 3.15–3.19. The data points for $t > 300$ were collected, and points for which $a_k(t) < 0.1$ were removed so that further calculations were not affected by the behaviour at very small amplitudes, which is easily influenced by noise. A curve of the form $a = a_0 \exp(\omega t)$ was fitted to the time series for each value of k and each value of g_{cc}, giving a dispersion curve $\omega(k)$ for each value of the coupling constant g_{cc}. These curves are plotted in figure 3.20.

Figure 3.20 shows several effects of varying the coupling constant: low-g_{cc} simulations of effectively miscible fluids show a negative growth rate, although it is very possible that this decay could be suppressed by running simulations at higher forcing rates, reducing the effect of diffusion at the interface compared to hydrodynamic effects. As g_{cc} is raised, the critical wave-vector k_c at which $\omega(k_c) = 0$ increases at first, due to reduced diffusivity. However, beyond $g_{cc} = 1$, the critical wave-vector decreases again: this is attributed to the increased surface tension (see figure 2.1) suppressing short-wavelength disturbances.

For $g_{cc} = 1.5$, the surface tension $\sigma = 0.025$; the mean interface velocity was measured to be 0.00989, suggesting a critical wave-vector of 0.69 ± 0.01, in comparison to the observed value of 0.25. Similar discrepancies exist for other values of g_{cc}. However, the shape and qualitative behaviour of the dispersion curves is as expected[126].

3.5 Conclusions

The 2D Shan-Chen lattice BGK model for multicomponent interacting fluids was modified in order to allow simulation of fluids in a Hele-Shaw cell. An oversight in previous, similar modifications, which caused inaccuracy and numerical instabilities at high shear rates or small cell widths, was corrected, and quantitative verification of the correction in a variety of scenarios was obtained. The 2D approximation of Hele-Shaw flow in equation 3.5 improves as the cell gap width h is reduced; the approximation permits simulation at significantly lower values of h than before. It was possible to reproduce a hydrodynamic interface instability of the form described by Saffman and Taylor. Dispersion relations were obtained for this instability, and while they showed qualitatively reasonable behaviour, further work is required for quantitative comparison with known results from experiment and theory.

The use of a "bottom-up" interaction scheme was found to complicate the examination of viscous fingering. The fingering effect involves the interplay of several different mechanisms which all depend on the surface tension, diffusivity, and viscosity of the fluid mixture, yet these properties emerge in a sometimes complicated way from the coupling constants of the Shan-Chen model; variation of a single parameter such as g_{cc} or τ_r immediately changes all three fluid properties, making it difficult to vary such properties independently. A better choice for future work might be the more "top-down" model of Swift *et al* , which has a more explicit connection to the hydrodynamic parameters in terms of which much discussion of the Saffman-Taylor effect is discussed in the literature. Alternatively, the Gunstensen-Rothman model provides a sharp interface, and has been improved to give reduced interfacial anisotropy and reduced spurious flow around interfaces[135]. However, the Hele-Shaw modification proposed here is independent of the interaction scheme, so that it should be possible to use it in conjunction with other schemes.

It should be noted that the effects of Hele-Shaw flow on fluid mixtures can be subtle and complicated. For example, Saffman and Taylor noticed that a thin film of displaced fluid can be left on the cell walls during viscous fingering experiments; a correction to the surface tension may also have to be made in stability calculations, due to interfacial variation in the direction perpendicular to the Hele-Shaw cell. None of these effects are accounted for in the model described here. Finally, the Hele-Shaw flow approximation is derived purely from macroscopic considerations. There has recently been some success in deriving lattice BGK models and improved boundary conditions for flow in very restricted geometries[136], by writing down a description of the geometrical and boundary constraints at the level of the continuum Boltzmann equation, rather than the Navier-Stokes equation; this can then be discretized to give a lattice Boltzmann model for numerical calculation. A similar approach may yield improved models for Hele-Shaw flow.

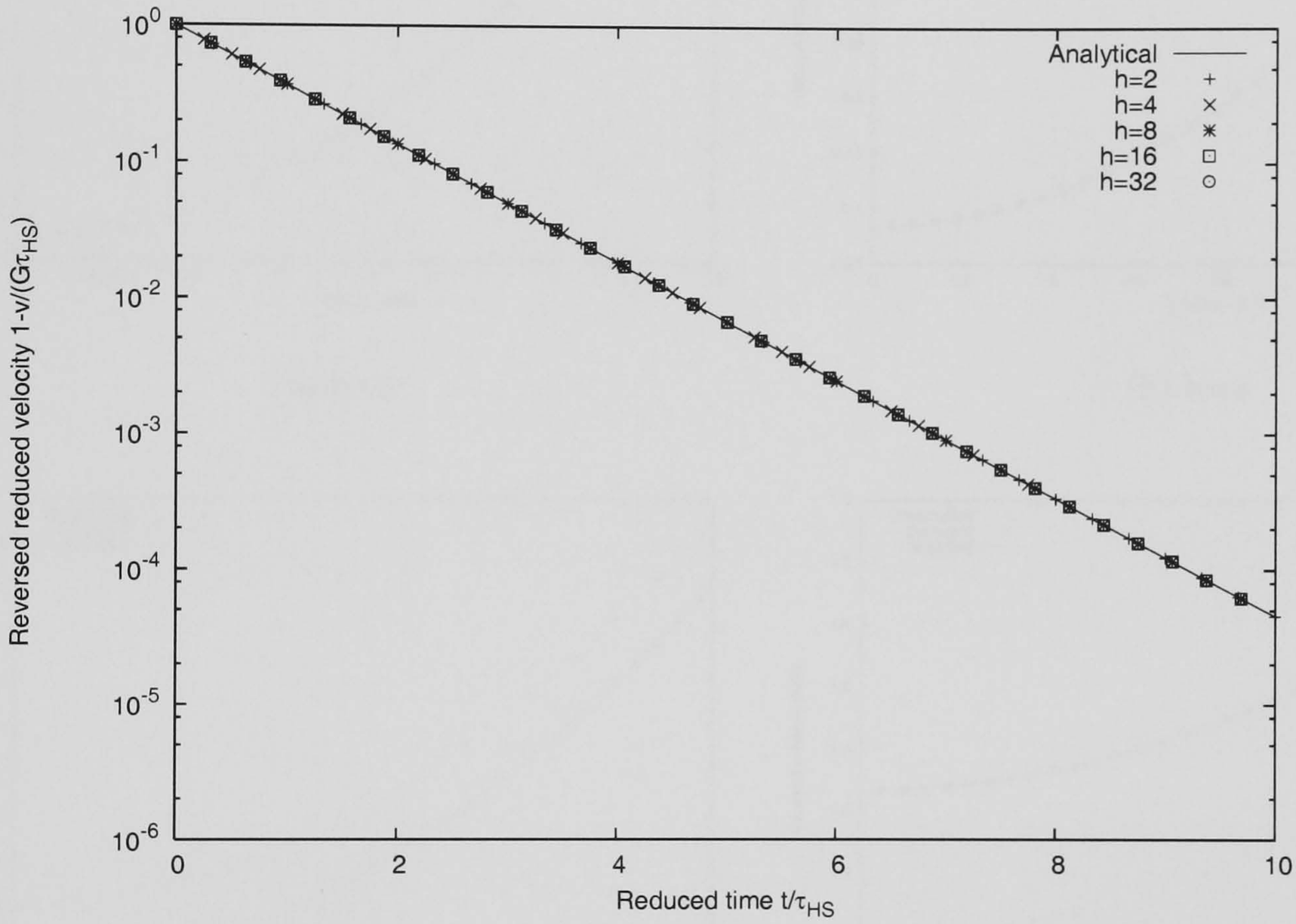
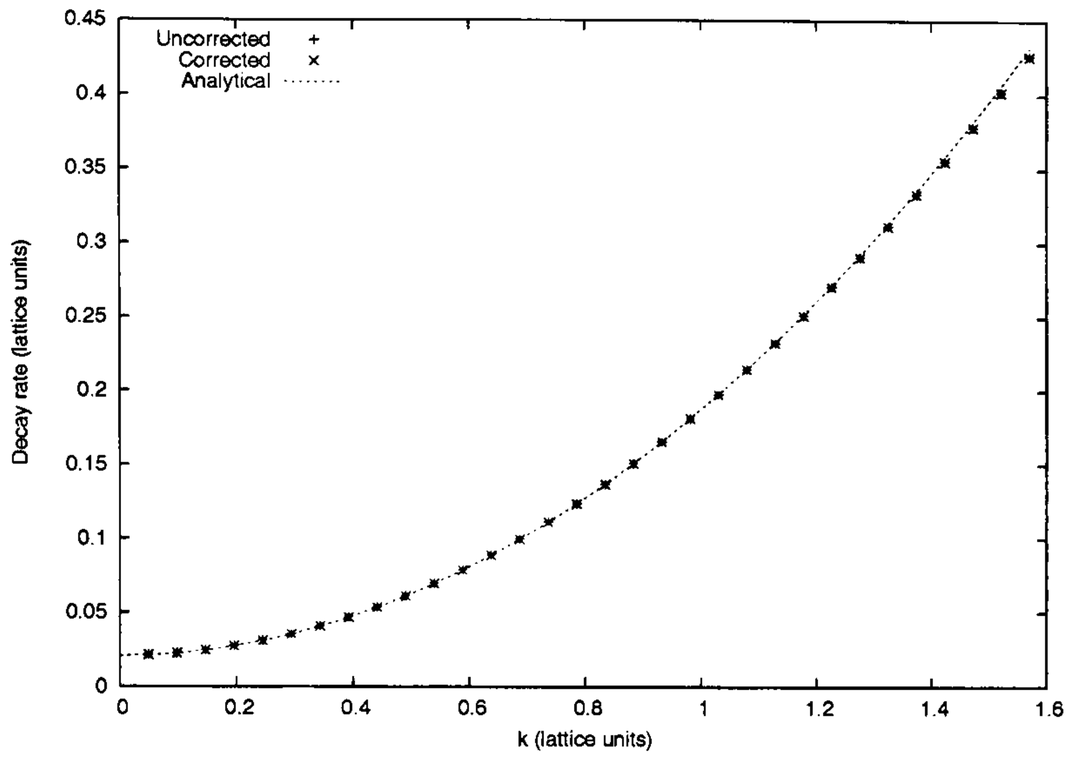
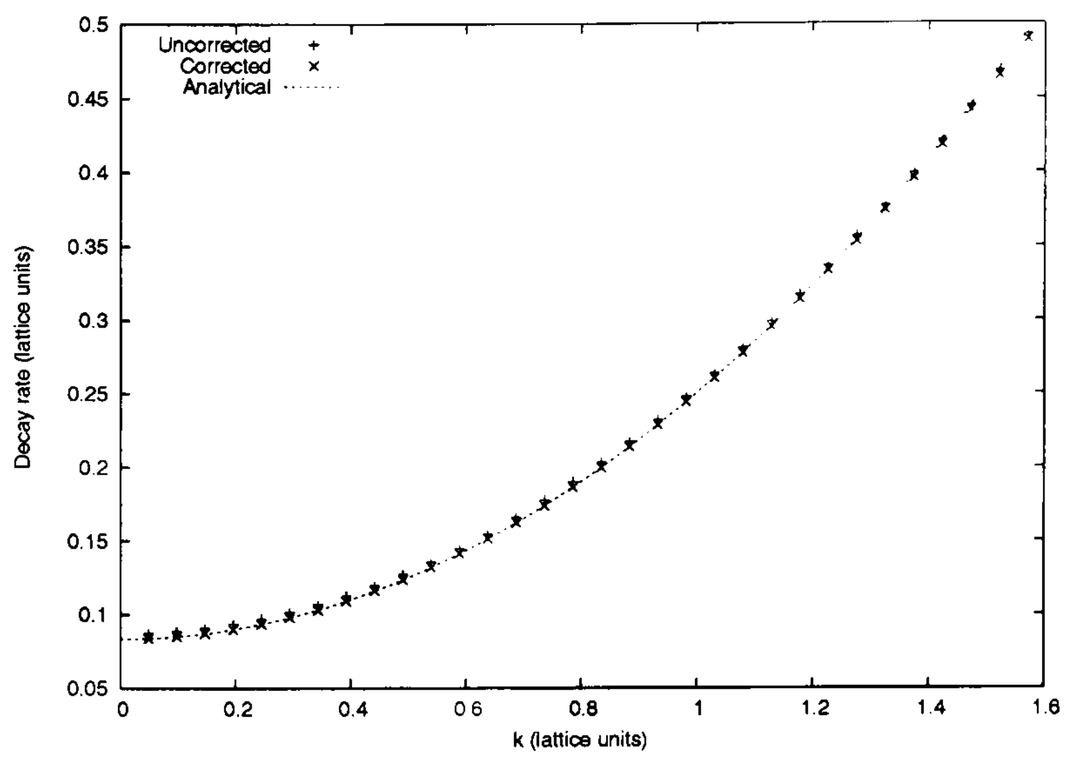


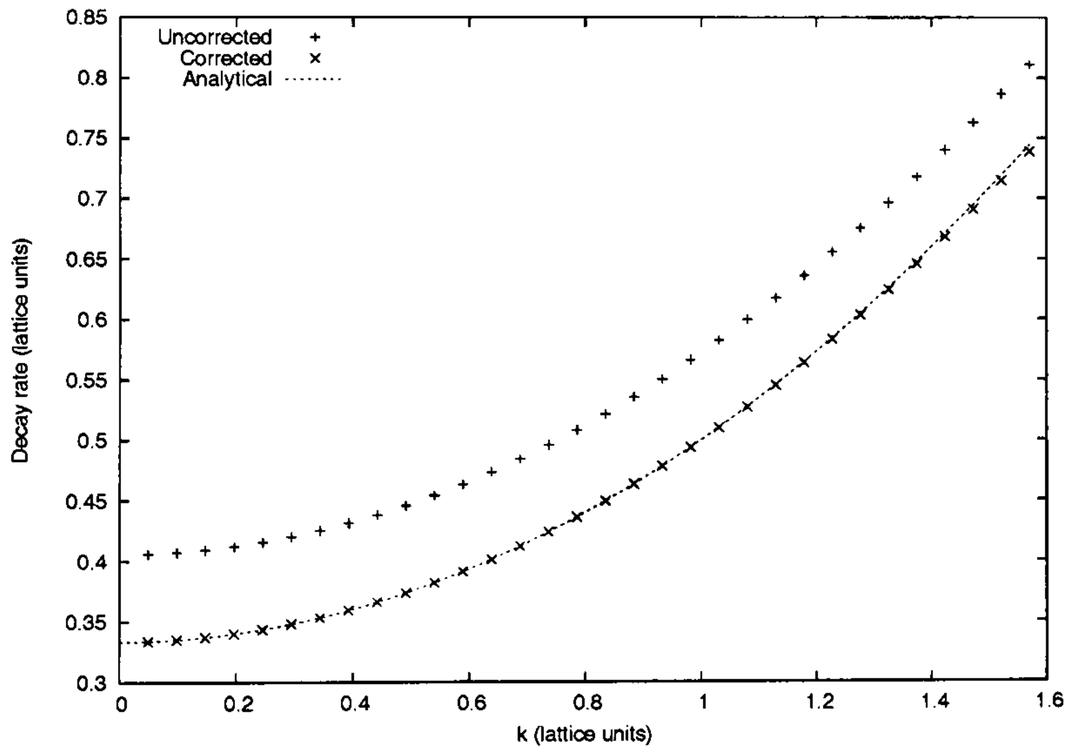
Figure 3.9: The velocity of fluid accelerating under gravity in a Hele-Shaw cell plotted against time, simulated using lattice BGK plus a drag term corrected to account for variation of the drag force during a single lattice BGK timestep.



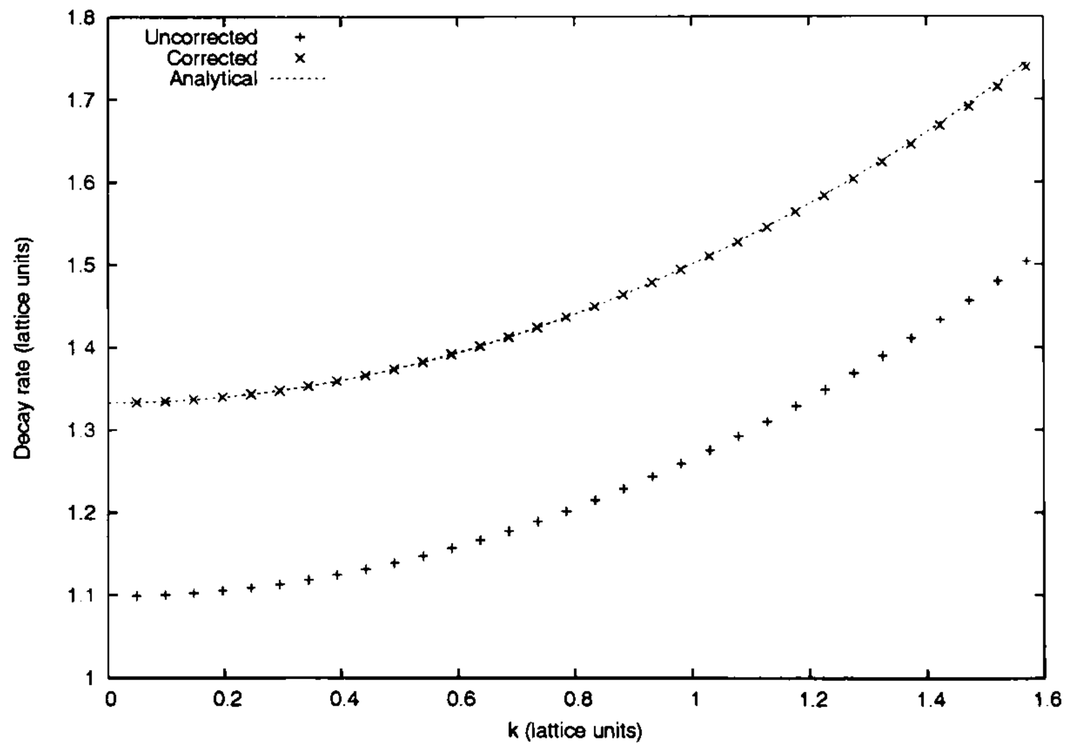
(a) h=8



(b) h=4



(c) h=2



(d) h=1

Figure 3.10: Dispersion relation for shear waves at different Hele-Shaw cell widths, showing decay rate ω against wave-vector k for the analytical result, and simulations with and without drag correction. Error bars are omitted, being smaller than the point markers.

Figure 3.11: A simulation of viscous fingering. Black regions correspond to fluid of high Darcy permeability (analogous to low viscosity); white regions correspond to fluid of low permeability or high viscosity.

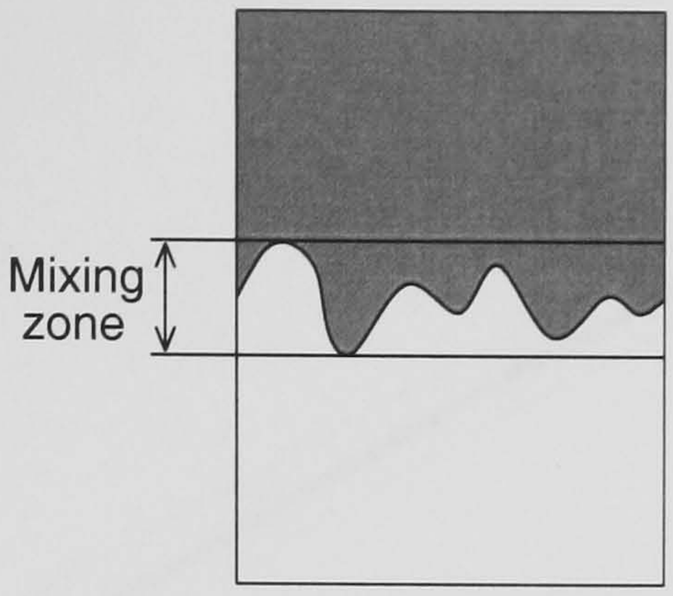


Figure 3.12: The width of the mixing zone is defined as the difference between the maximum and minimum depths of the curve $\phi = 0$.

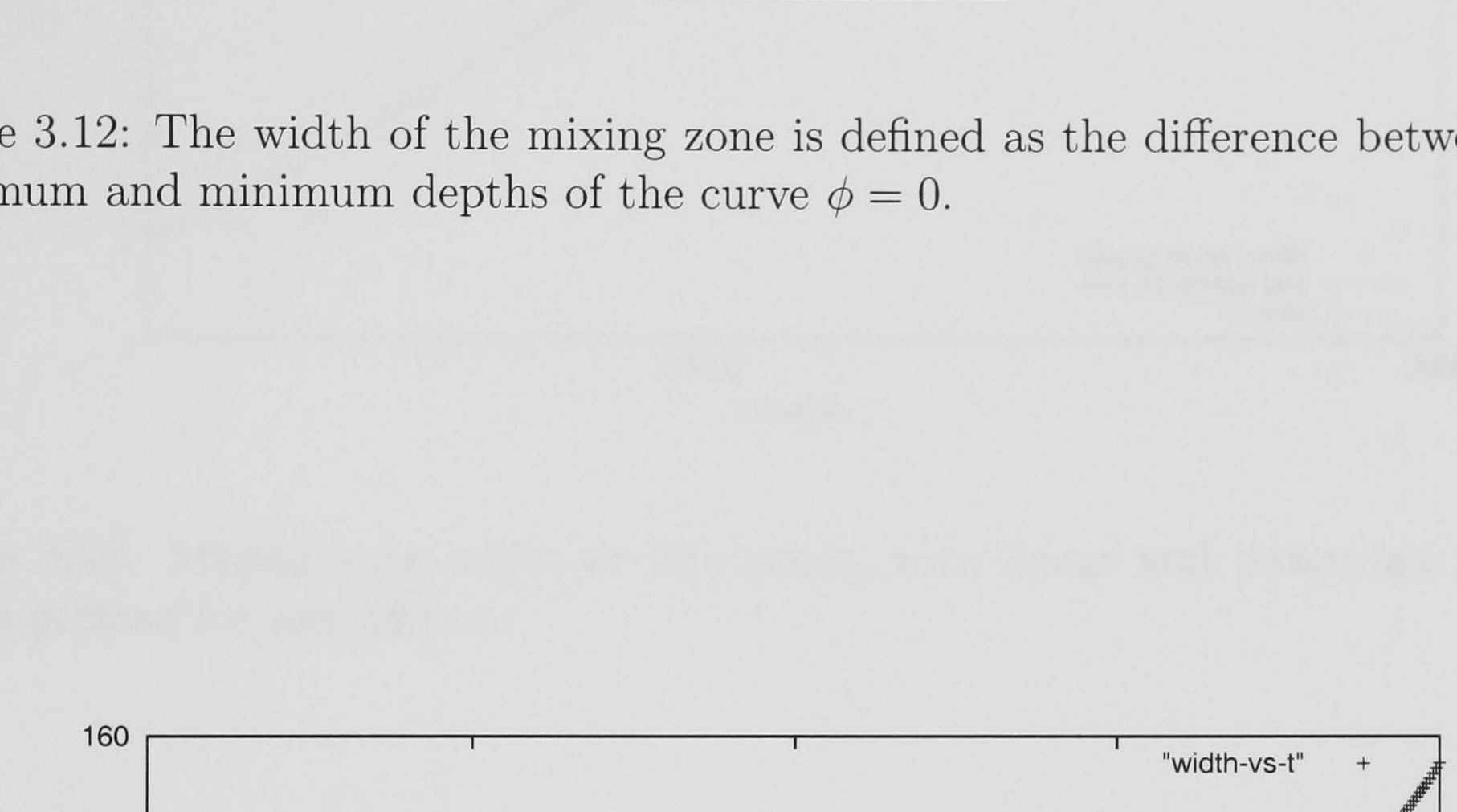


Figure 3.13: The width of the mixing zone during a viscous fingering simulation, plotted against time.

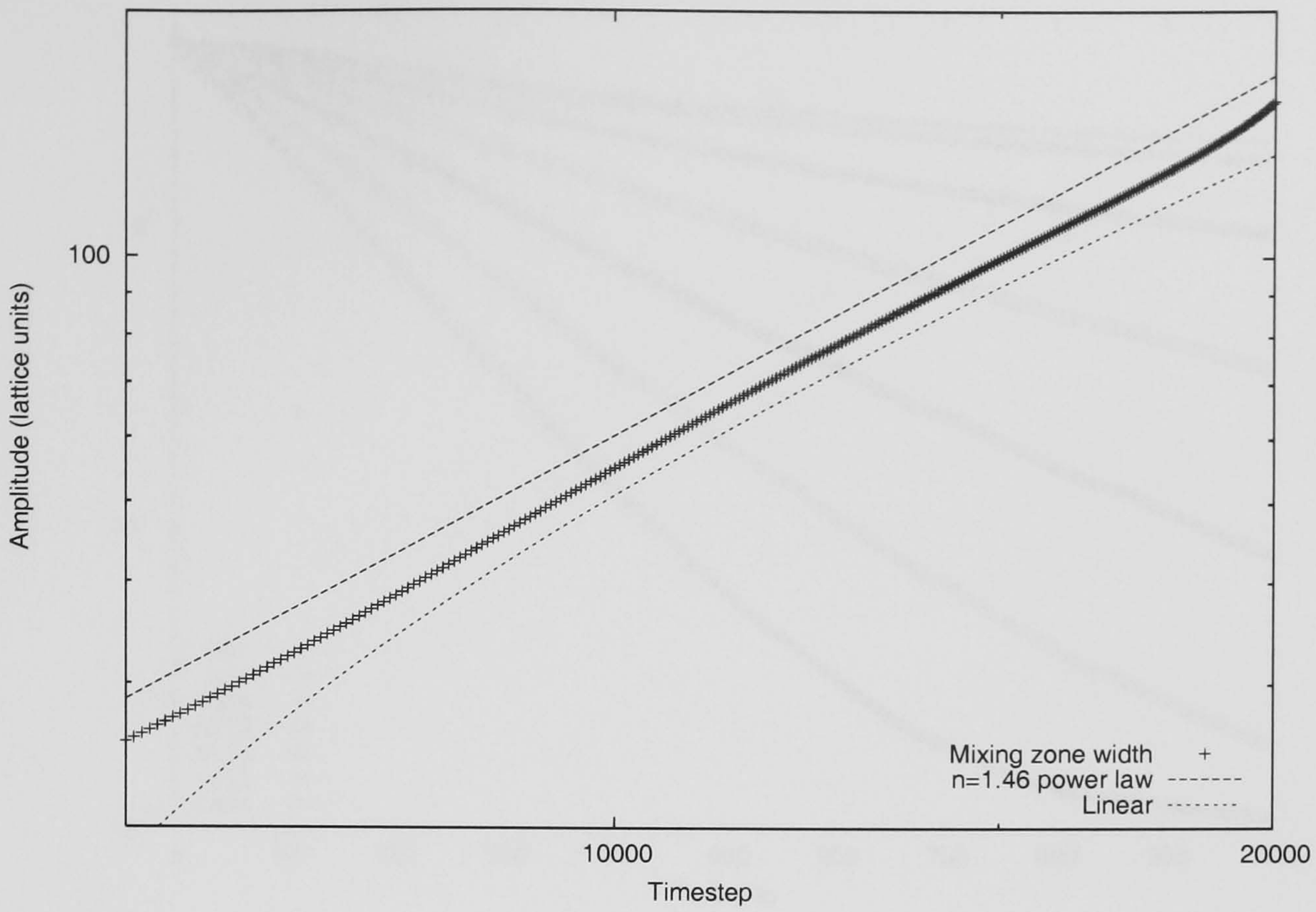


Figure 3.14: Mixing zone width at late times, with linear and power-law growth curves plotted for comparison.

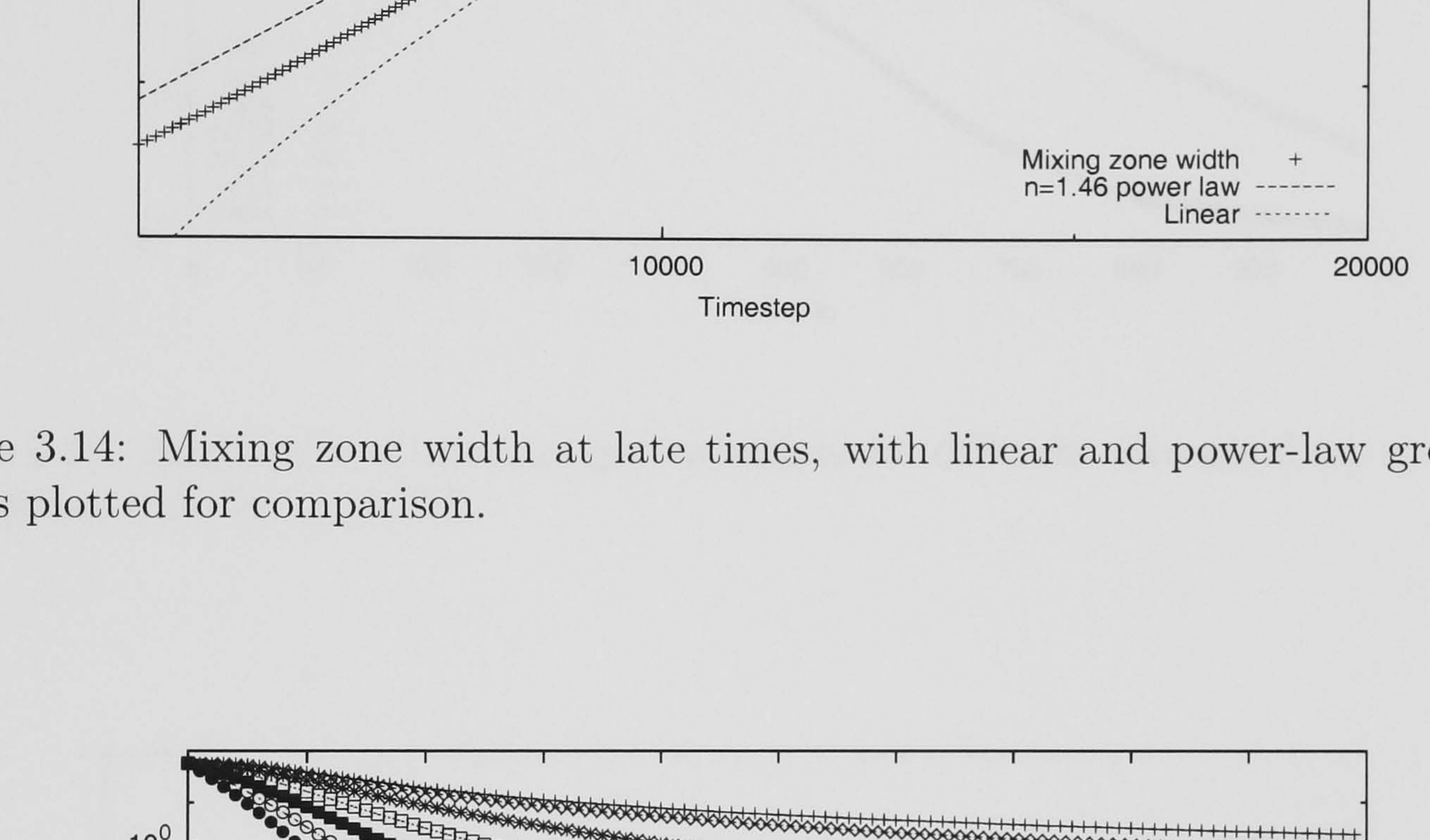


Figure 3.15: Amplitude of interface perturbations for different wavenumbers n, plotted against time for $g_{cc} = 0$.

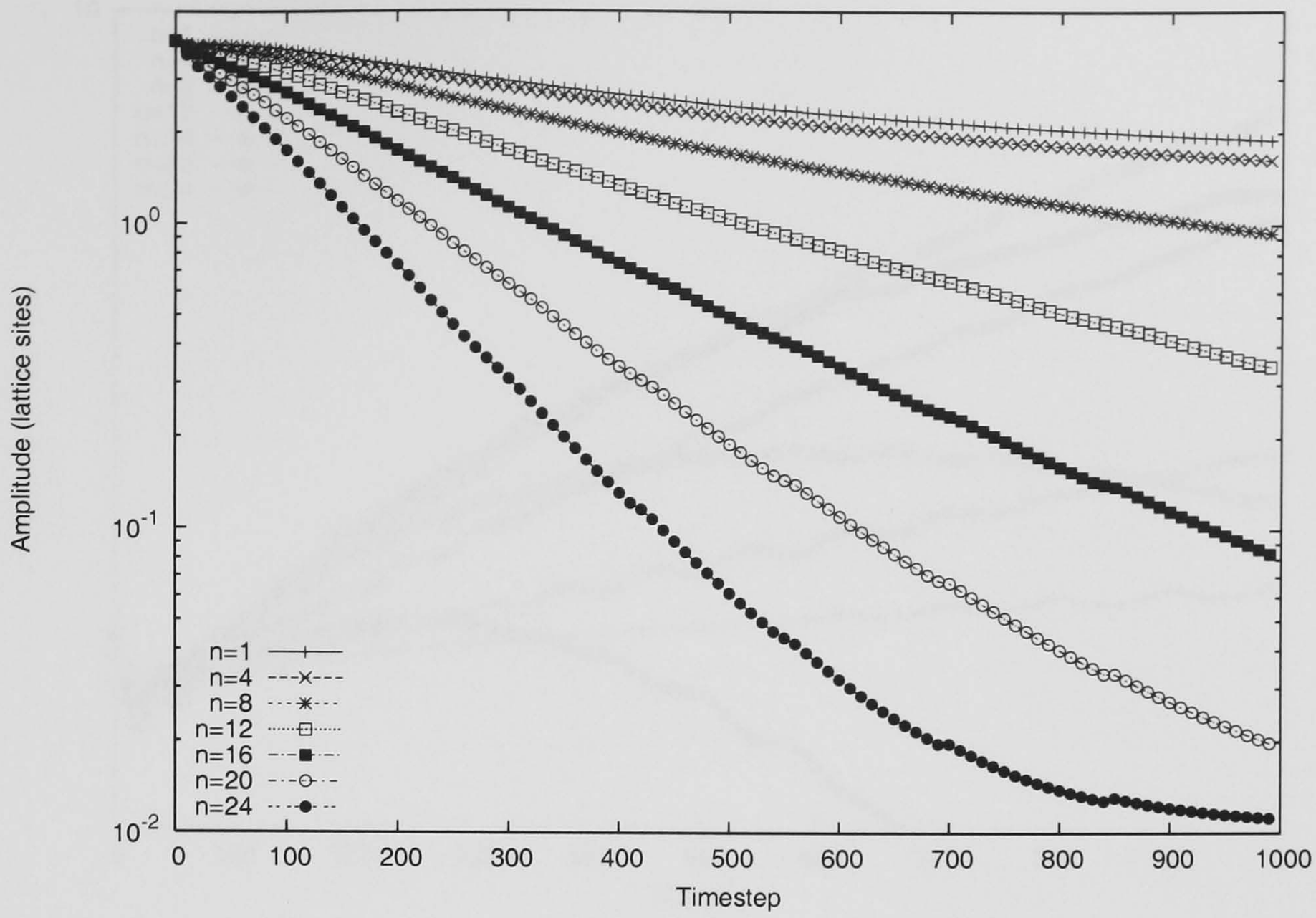


Figure 3.16: Amplitude of interface perturbations for different wavenumbers n, plotted against time for $g_{cc} = 0.5$.

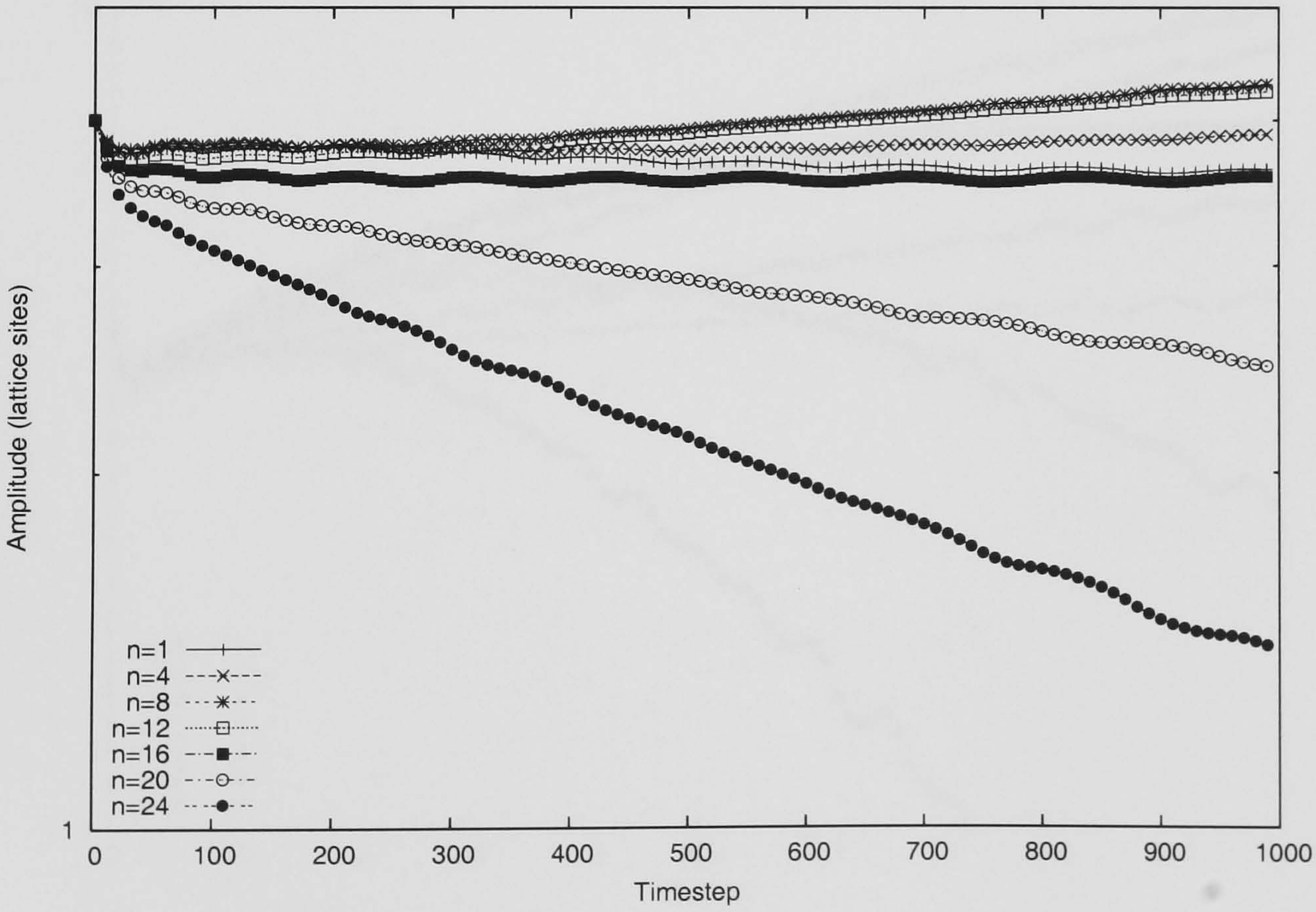


Figure 3.17: Amplitude of interface perturbations for different wavenumbers n, plotted against time for $g_{cc} = 1.0$.

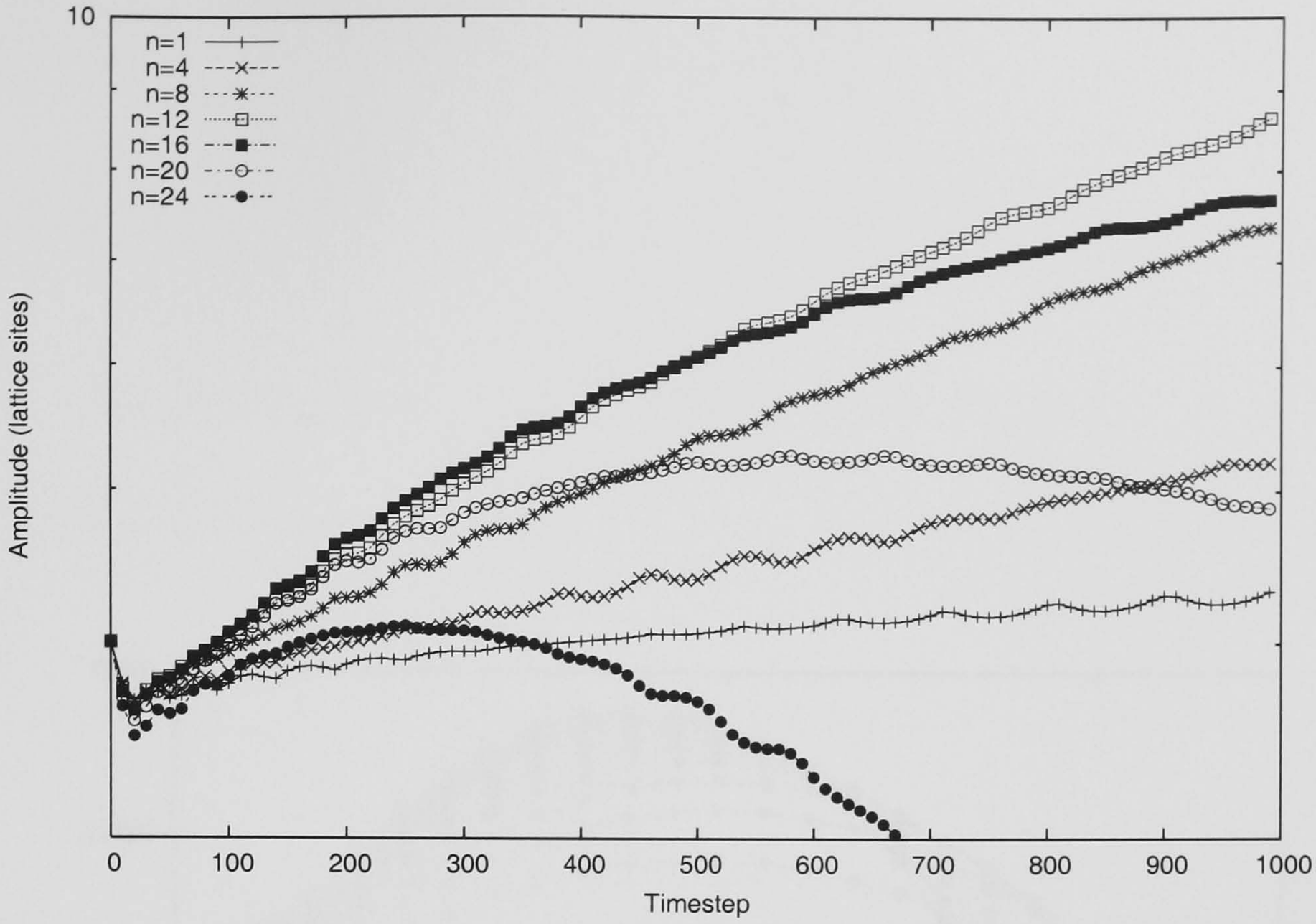


Figure 3.18: Amplitude of interface perturbations for different wavenumbers n, plotted against time for $g_{cc} = 1.5$.

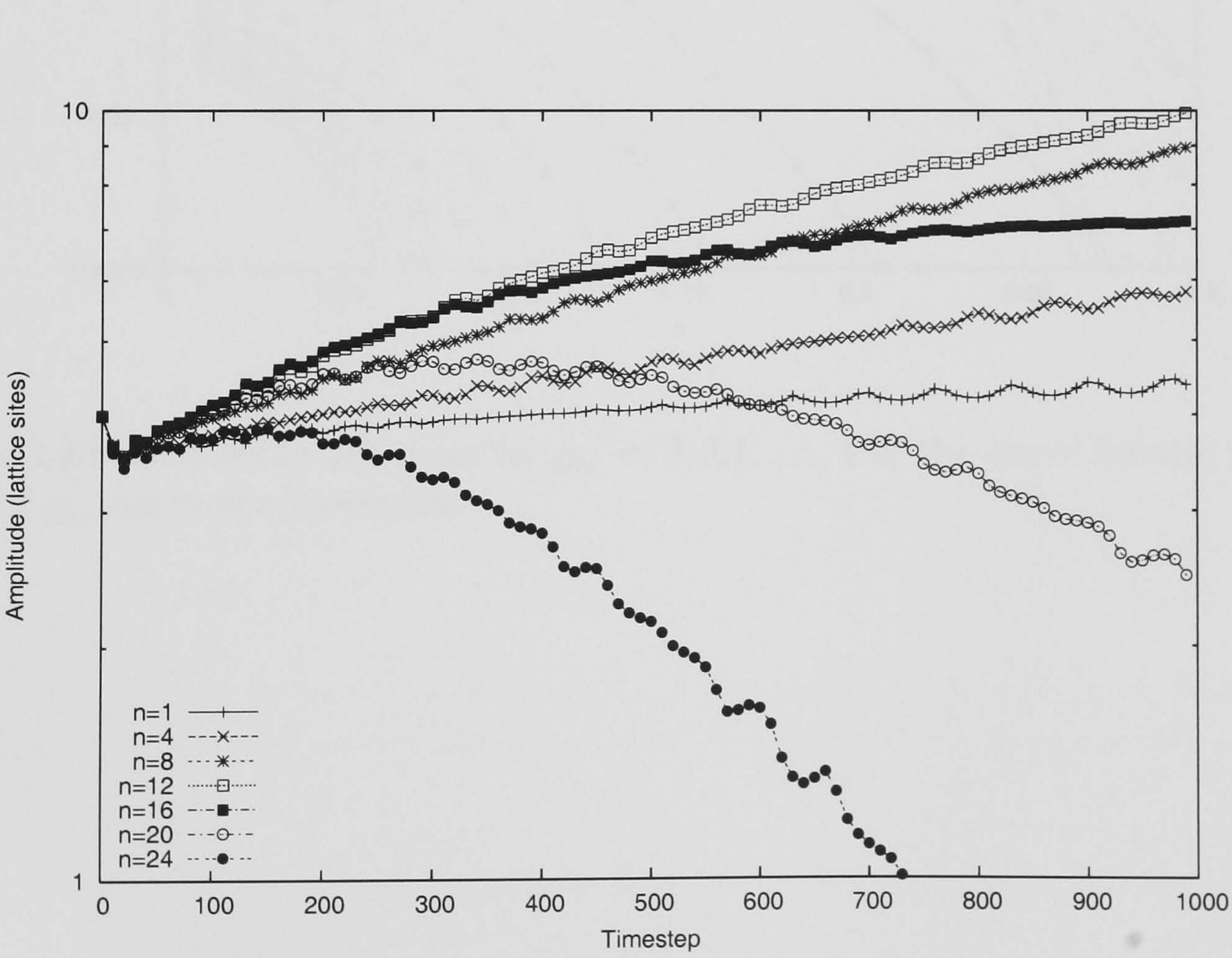


Figure 3.19: Amplitude of interface perturbations for different wavenumbers n, plotted against time for $g_{cc} = 2.0$.

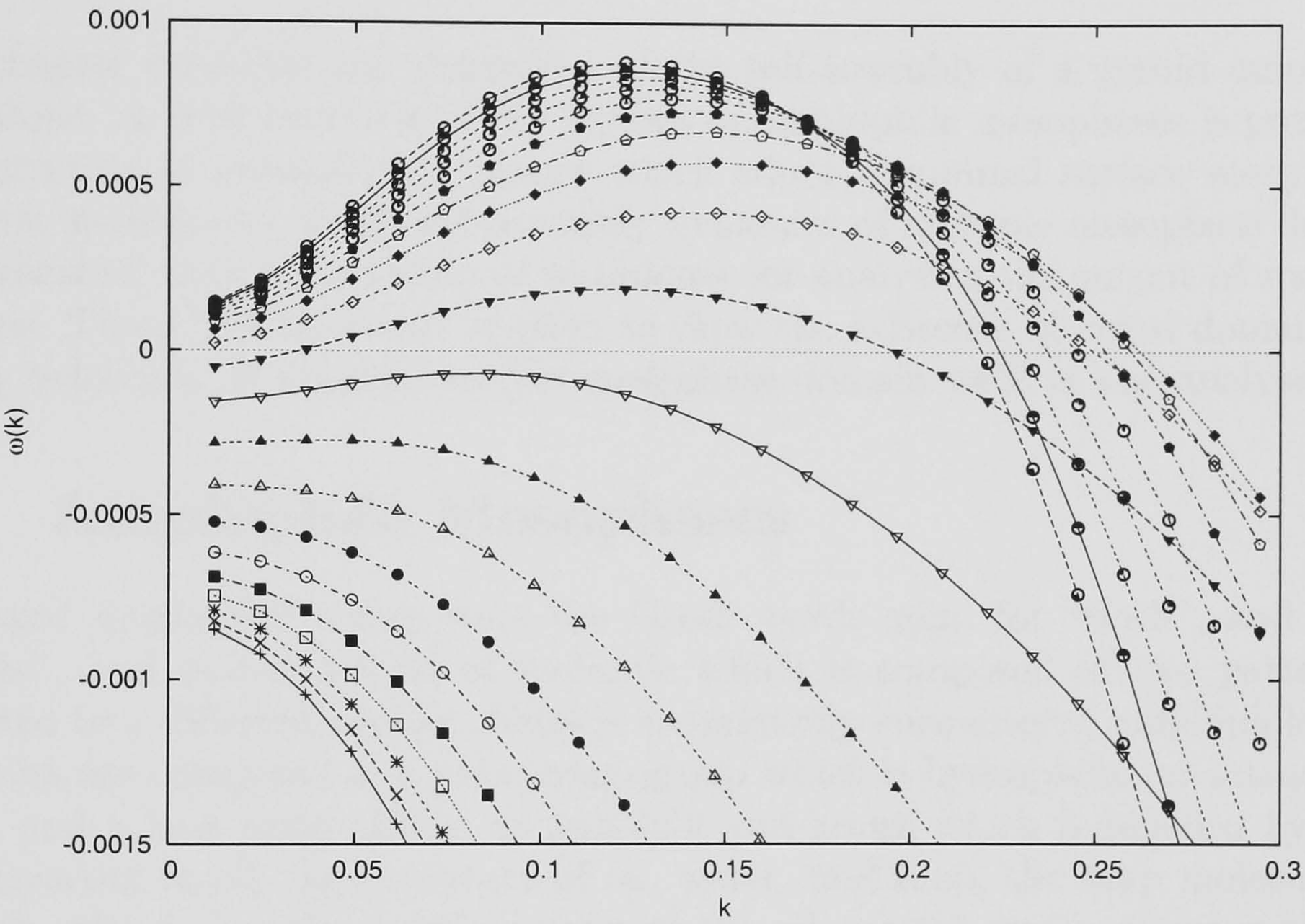


Figure 3.20: Dispersion relations for $g_{cc} = 0, 0.1, \ldots, 1.9$; the curve for the lowest value of g_{cc} starts the lowermost.

Chapter 4

Self-assembly of the Gyroid mesophase

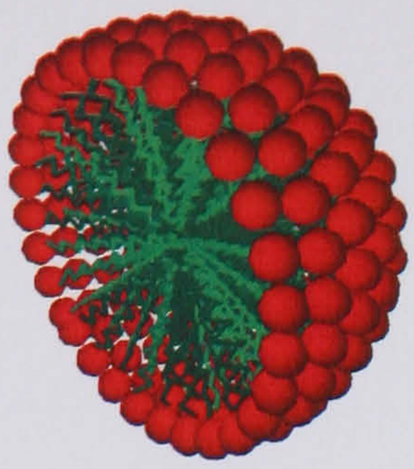
This chapter describes the simulation of the self-assembly of a gyroid amphiphile mesophase. A brief overview of the physics of amphiphile mesophases is presented, concentrating in particular on phases which adopt a minimal surface morphology. The first simulations of the self-assembly dynamics of multiple mesophase domains are presented, with a discussion of techniques for analysing the output of such simulations. These techniques are applied to show the existence of chiral domains; the scaling behaviour of minimal surface mesophase domain walls is also analysed.

4.1 Amphiphile Mesophases

The word *amphiphile* comes from the Greek words αμφι, for "both", and φιλειν, "to like", and means a kind of molecule which is composed of two parts, each attracted to a different species. Soap is a commonly encountered amphiphile: soap molecules are composed of a polar head-group which is hydrophilic, or attracted to water, and a long hydrophobic or lipophilic tail group which is repelled by water and attracted to oil. In a mixture of oil, water, and soap, the soap molecules are strongly attracted to the interface between the oil and the water, allowing them to sit in the minimal-energy configuration with the head-group facing towards water and the tail group facing into the oil. This tendency to migrate towards interfaces is why such molecules are commonly termed *surfactants*, or surface-active agents. Surfactants have immense industrial importance, mainly due to their tendency to sit at interfaces and thereby reduce surface tension: this is the principle by which all detergents work.

While a solution of surfactant in water is easily prepared in the kitchen sink, such solutions show very rich and complicated behavior, summarised by, eg, Langevin[137], Gompper and Schick[70], or the reviews of Seddon and Templer[138, 139]. This behaviour depends on many variables, such as temperature, surfactant volume fraction, non-polar chain length, polar head-group strength (which in turn depends on factors such as pH and ionic concentration), and the form of the interaction between tailgroups, between head and tailgroups, and between headgroups.

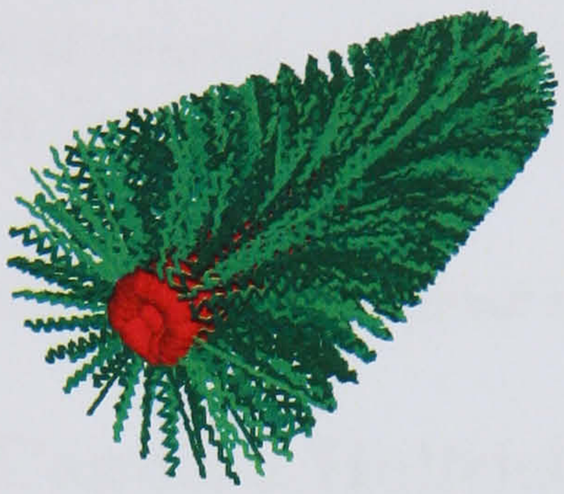
The very simplest models of surfactant behavior[137] consider two effects: minimization of the area of the interface between oil-like and water-like regions, due to the hydrophobic effect, and geometrical effects arising from the relative strength and bulk of the head and tail groups.



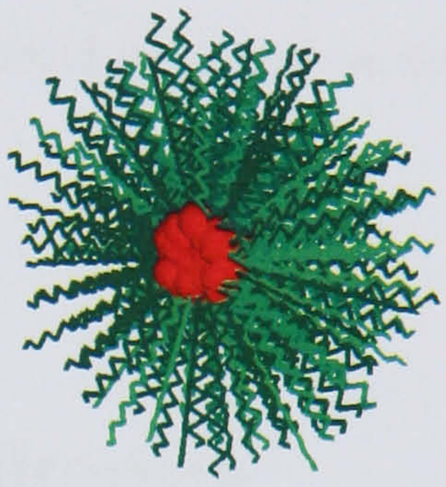
(a) Spherical micelle

(b) Wormlike micelle

(c) Lamella



(d) Inverse wormlike micelle



(e) Inverse spherical micelle

Figure 4.1: Simple surfactant morphologies.

At very low concentrations, free surfactant molecules may be found as monomers, on their own. At a certain point, either the excess surfactant will precipitate (the Krafft point[140]), or the surfactants will start to assemble into aggregate structures, in which case the point is called the Critical Aggregation Concentration (CAC). The aggregates are in equilibrium with the monomers, whose concentration remains close to the CAC.

If the tail group is relatively compact, then packing surfactant molecules together produces a nett curvature towards the water, and they cluster into groups called *spherical micelles*, as shown in figure 4.1(a), allowing the tail groups to congregate without touching any water.

For larger tail groups, it becomes energetically more favourable for the surfactants to form long cylindrical *wormlike micelles* (Figure 4.1(b)). Wormlike micelle phases are many and varied: the micelles may be stacked into an ordered hexagonal phase, or they may form interpenetrating branched structures. Such phases exhibit a rich non-Newtonian rheology, since shear forces may align, disentangle, or break micelles; understanding this typically mesoscale problem is of much industrial interest.

For even larger tail groups, the easiest way to shield the tail groups from the surrounding water is for the surfactant molecules to form lamellar phases, assembling into flat sheets (Figure 4.1(c)).

Surfactants with larger tail groups will form inverse wormlike micelles (figure 4.1(d)) and inverse spherical micelles (figure 4.1(e)), with the tail groups pointing

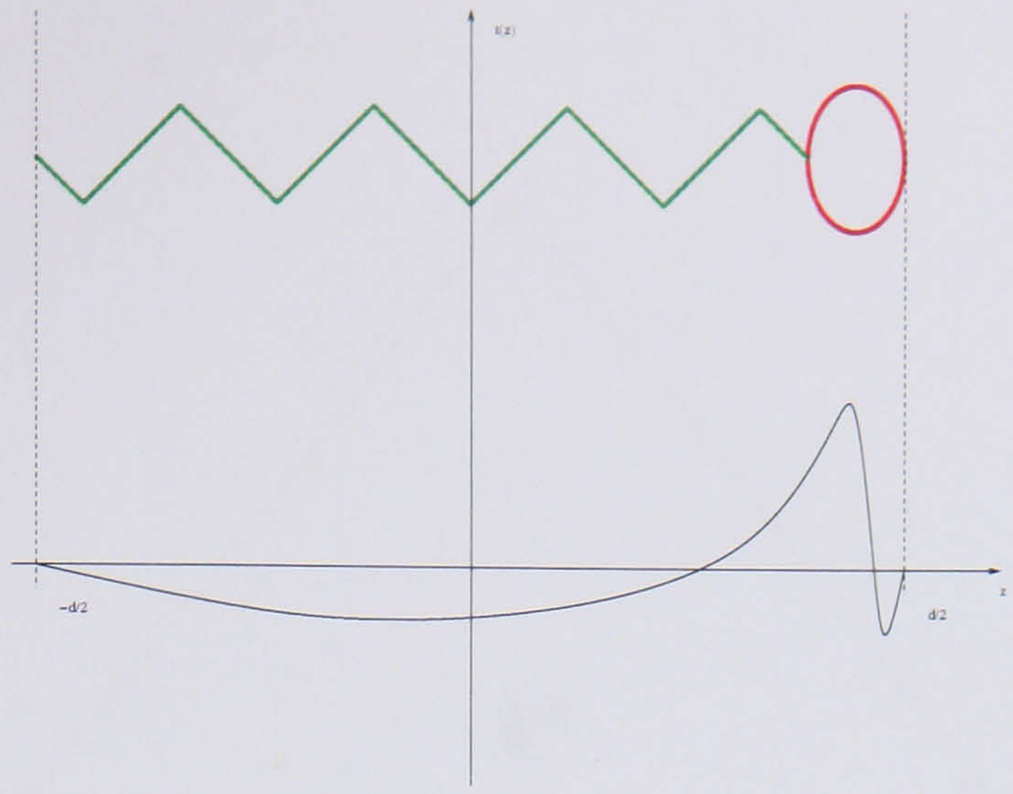


Figure 4.2: Lateral tension as a function of position along a surfactant molecule (after Seddon and Templer[138]).

outwards, if immersed in a non-polar solvent.

4.1.1 The Canham-Helfrich Hamiltonian

A surfactant molecule in a monolayer will have a variety of lateral forces acting on it, as summarised in figure 4.2. Repulsion between tail groups and between head groups acts to push molecules apart; surface tension effects act to pull them together. The effects of these forces can often be difficult to separate, and the nature of the individual forces is still poorly understood[138]; it is therefore simpler to describe the system in terms of the nett effects of the forces on the interfacial surface: specifically, the effects on the curvature of the surface.

Consider a point on a surface, with normal $\hat{\mathbf{n}}$ as shown in figure 4.3; a plane coincident with $\hat{\mathbf{n}}$ will intersect the surface in a curve q, which has radius of curvature r and curvature $c = 1/r$ at that point. If the plane is rotated about $\hat{\mathbf{n}}$, the radius of curvature will have maximum and minimum values r_2 and r_1, with sign defined according to $\hat{\mathbf{n}}$; the corresponding maximum and minimum curvatures at that point are called the principal curvatures c_1 and c_2.

The mean curvature H is defined as being proportional to the sum of the principal curvatures:

$$H \doteq \frac{1}{2}(c_1 + c_2)\,. \tag{4.1}$$

The Gaussian curvature K is defined as the product of the principal curvatures:

$$K \doteq c_1 c_2. \tag{4.2}$$

Consider a hypothetical symmetric amphiphile, whose head and tail groups are of the same size and exert the same magnitude of forces. At an interface, the ideal configuration of amphiphile molecules is planar (figure 4.4(a)): by symmetry, there is no innate preference for the interface to curve in one direction or another.

If, however, the tail groups dominate, and push one another apart more strongly than the head groups do, then there will be a tendency for the interface to curve to accommodate this, as shown in figure 4.4(b). The interface will then have a *spontaneous mean curvature* H_0.

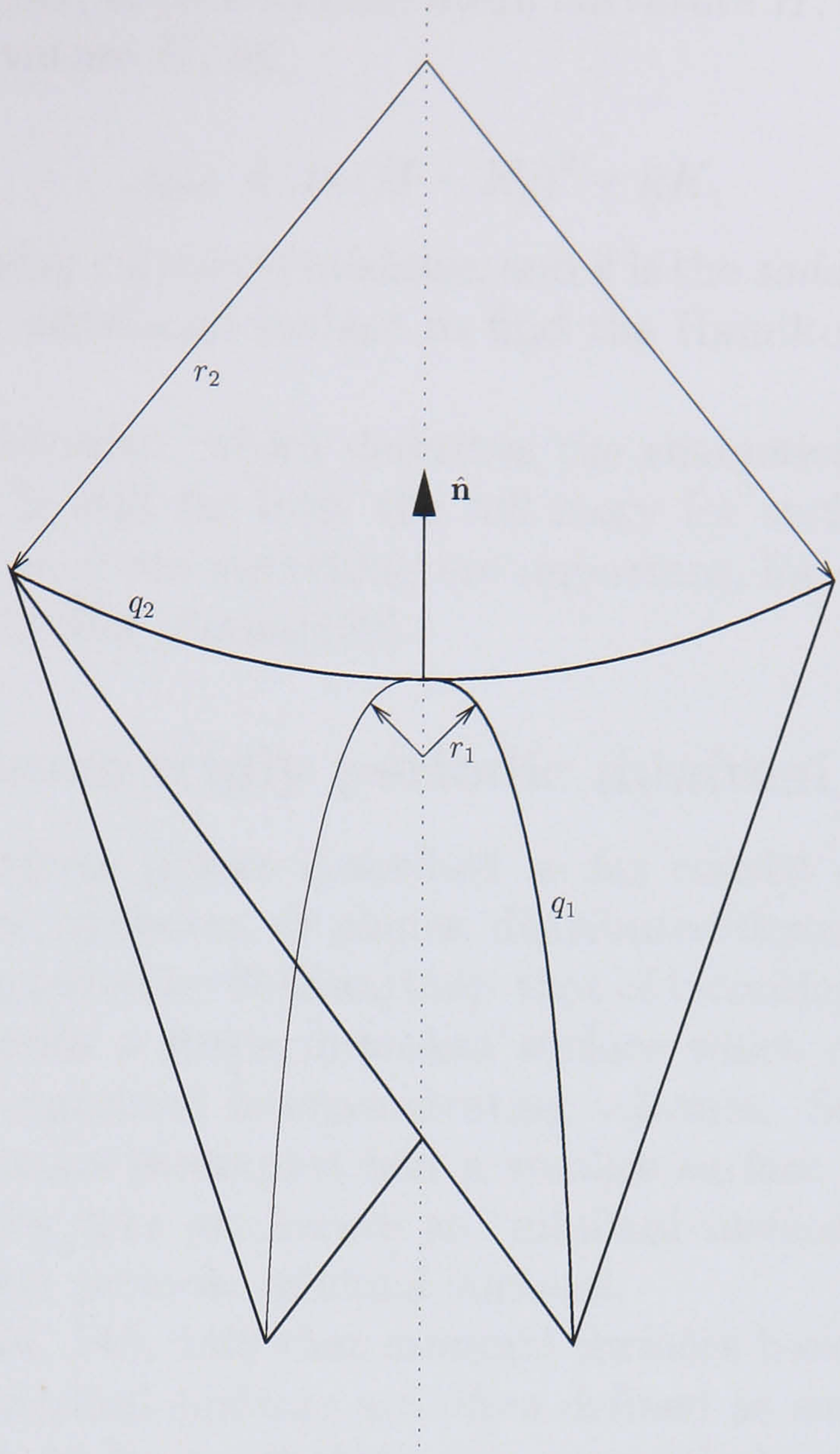
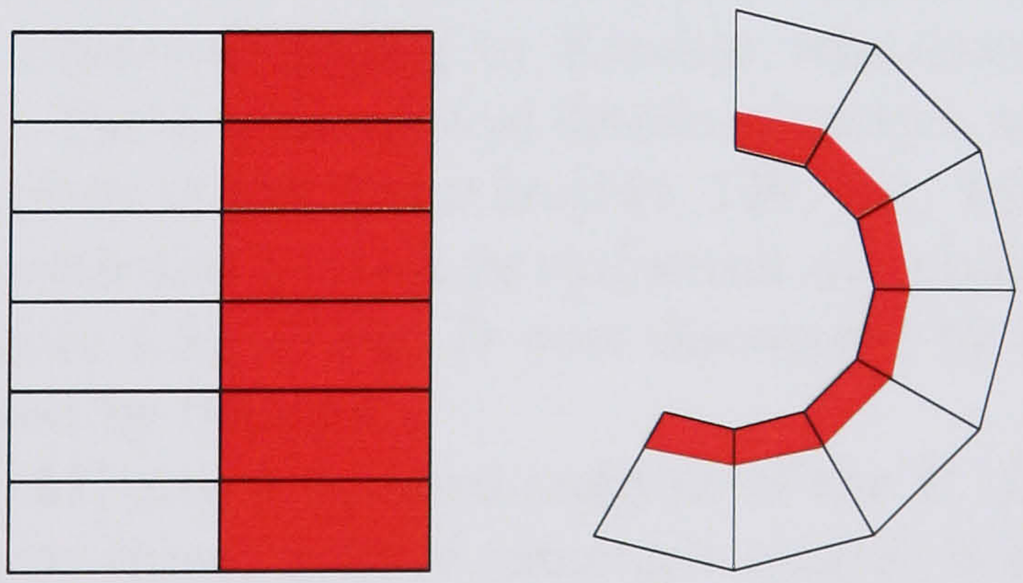


Figure 4.3: Definition of the principal curvatures at a point on a surface.



(a) Zero spontaneous mean curvature

(b) Nonzero spontaneous mean curvature

Figure 4.4: Amphiphile structure for zero and nonzero mean curvatures.

The free energy cost of curvature at a point on an interface was characterised by Canham and Helfrich[141] in terms of the mean curvature H, spontaneous curvature H_0, and Gaussian curvature K, as

$$g_{\text{curv}} = 2\kappa \left(H - H_0\right)^2 + \bar{\kappa} K, \tag{4.3}$$

where κ is called the *splay curvature modulus*, and $\bar{\kappa}$ is the *saddle splay modulus*. This is integrated over the interfacial surface to find the Hamiltonian for the complete system.

The Helfrich Hamiltonian, which describes the energetics of a fictional surface of zero thickness[142] is still far from the full story for surfactants; packing frustration and hydrocarbon chain stretching are important, for example in the inverse hexagonal wormlike micellar phases[139].

4.1.2 Bicontinuous triply periodic minimal surface phases

The amphiphilic aggregate phases described so far consist of many disconnected objects, such as spheres, cylinders, or planes, distributed throughout space. Another possibility was first suggested by Scriven[143]: that of bicontinuous structures, where the surfactant layer forms a single unbroken surface which divides space into two continuous, multiply connected interpenetrating volumes. Scriven argued that, in certain cases, bicontinuous structures had a smaller surface area than aggregates. Surfaces which minimize area are known as "minimal surfaces"; ones which repeat through space are called periodic minimal surfaces.

It can be shown[144, 145, 146] that minimal surfaces have zero mean curvature everywhere (in fact, minimal surfaces are often defined as surfaces with zero mean curvature, rather than in terms of their area properties), so that $c_1 = -c_2$, and therefore the Gaussian curvature $K = c_1 c_2$ is negative everywhere.

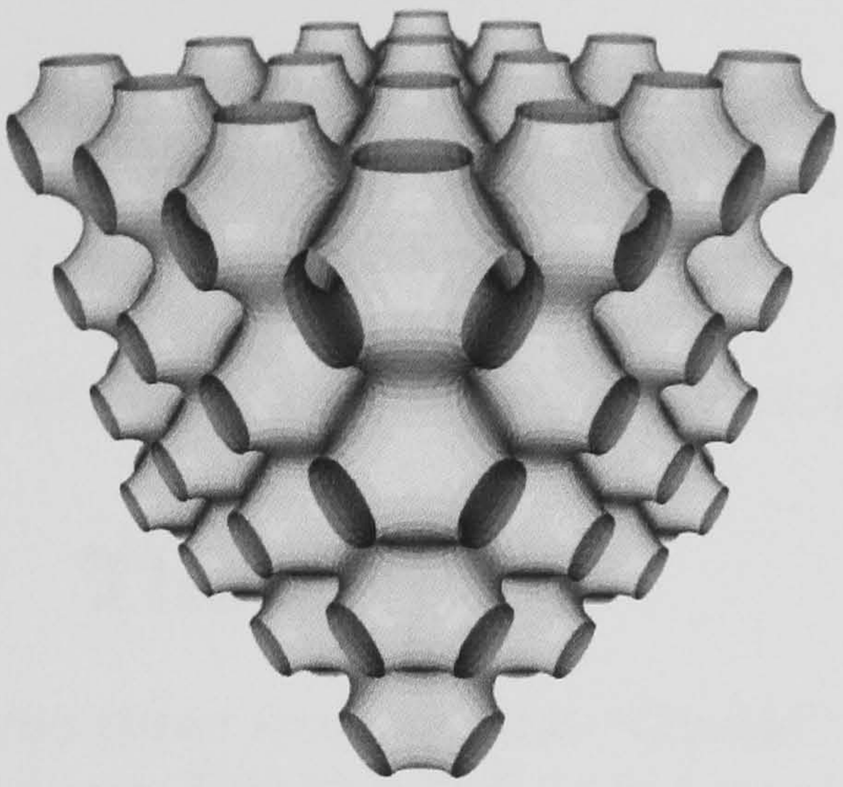
A Triply Periodic Minimal Surface (TPMS) is a minimal surface with cubic symmetry, repeating in the X, Y, and Z directions.

A brief history of the TPMS is given by U. S. Schwarz and G. Gompper[147]: until 1970, only five TPMS were known, discovered by H. A. Schwarz and his students in the late 19th century. In 1970, Schoen[148] discovered twelve more, describing them in a NASA technical report. Little attention was paid to the new surfaces, until their existence was rigorously proven by Karcher, who discovered the existence of others in addition[149]. Further theoretical details of various aspects of periodic minimal surfaces are described in the literature[144, 150, 151, 149, 152, 153, 154, 155].

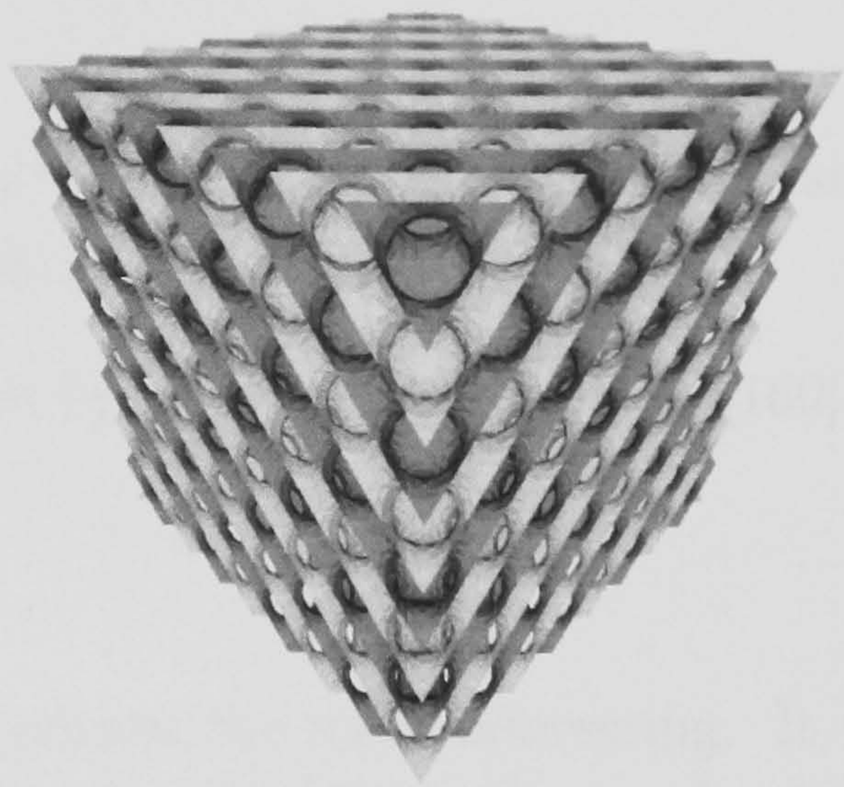
Three surfaces of particular interest in surfactant morphologies are called the P, D, and G surfaces (figure 4.5). P and D were discovered by H. A. Schwarz; G, the "gyroid", was discovered by Schoen.

Fogden and Hyde[157] give a detailed analysis of the P, D, and G surfaces. An analytical form for these three surfaces (amongst others) is known in terms of the Enneper-Weierstrass representation[158, 157], which maps the fundamental patch of each surface into the complex plane. It can be shown[158] that the P, D, and G surfaces all have the same Weierstrass representation except for a single parameter θ, called the Bonnet angle. Variation of θ maps the surfaces on to one another, and is called a Bonnet transformation.

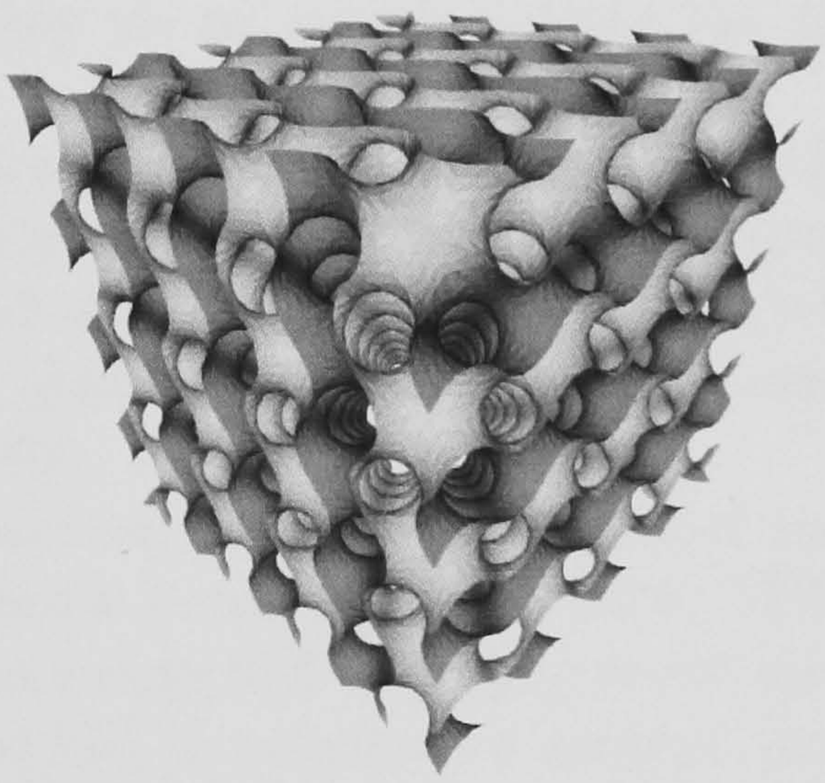
The Enneper-Weierstrass representation of these surfaces is unfortunately a little cumbersome to work with, since it involves several elliptic integrals. Nonetheless, it has been used to derive many properties of the surfaces, such as the area, Euler



(a) The P, or "Plumber's Nightmare" Surface



(b) The D, or "Diamond" surface, so called because each labyrinth has the same structure as the carbon atoms in a diamond crystal[156].



(c) The G, or "Gyroid" surface

Figure 4.5: Three common cubic triply periodic minimal surfaces.

characteristic, and Gaussian curvature, per unit cell. These have also been derived numerically for the gyroid and related surfaces of constant mean curvature[159], by discretizing a surface and permitting it to evolve, under volume constraints, to $H = 0$.

For many purposes, particularly visualization, the *P*-*D*-*G* surface family can be represented by a nodal approximation, where the surface is defined as the region $\Xi(\mathbf{r}) = 0$, for the functions $\Xi(\mathbf{r})$ given in equations 4.4–4.6.

$$\Xi_{\mathrm{P}}(\mathbf{r}) = \cos x + \cos y + \cos z \tag{4.4}$$

$$\Xi_{\mathrm{D}}(\mathbf{r}) = \sin x \sin y \sin z + \sin x \cos y \cos z + \cos x \sin y \cos z + \cos x \cos y \sin z \tag{4.5}$$

$$\Xi_{\mathrm{G}}(\mathbf{r}) = \cos x \sin y + \cos y \sin x + \cos z \sin x \tag{4.6}$$

More accurate nodal representations are given by Schwarz and Gompper[160].

4.2 The Gyroid

Of these three surfaces, the "Gyroid" *G* is perhaps the most interesting. It is the only known TPMS which is balanced (that is, the two labyrinths can be mapped onto each other through a Euclidean transformation) while containing no straight lines; it is also the only known TPMS composed entirely from triple junctions.

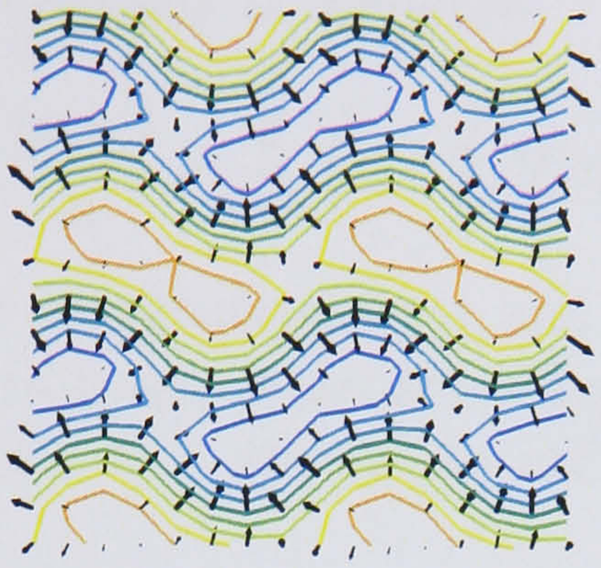
The gyroid has symmetry group I$a\bar{3}$d; the unit cell consists of 96 copies of a fundamental surface patch, related through the symmetry operations[158] of this space group. Channels run through the gyroid labyrinths in the (100) and (111) directions; passages emerge perpendicular to any given channel as it is traversed, the direction at which they do so gyrating down the channel, giving rise to the "gyroid" name[159].

The labyrinths are chiral, so that the channels of one labyrinth gyrate in the opposite sense to the channels of the other, as seen in Figure 4.6(c). Looking down the (111) direction of a gyroid shows a distinctive "wagon wheel" pattern[161] (Figure 4.6(d)), which has been observed experimentally in transmission electron micrographs of gyroid phases[162].

Gyroids have been observed in many experimental systems, and are usually regarded as the most commonly occurring of the cubic TPMS geometries. They have been seen in triblock copolymers[162] and lipid-water mixtures[138, 139, 163, 164]. Attempts have been made[165] to construct a ceramic nanostructured film with the gyroid morphology through the use of a self-assembling polymer template, raising the interesting possibility of fabricating chirally selective porous media. The possibility of using TPMS morphologies for photonic crystals is under active investigation[166].

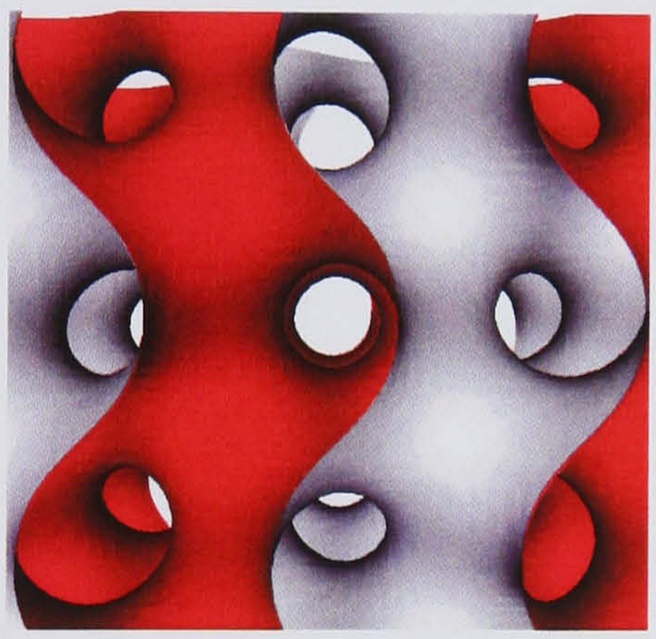
There have been hints that lyotropic liquid crystals may play a part in biological processes such as lipid digestion[167, 168], and offer insights into cell membrane properties and dynamics[169, 170, 171]. Donnay and Pawson[172] suggested that periodic minimal surfaces could be found in nature, and pointed at the microscopic structure of sea urchin skeletons[173] as a possible example; micrographic evidence has since emerged to suggest that cubic TPMS phases may be present in certain plant cell organelles, and Landh[174] suggests, with micrographic evidence. that gyroid structures may exist in the endoplasmic reticulum.

Schwarz and Gompper examined[160, 175] several TPMS morphologies, and showed that, for oil-water symmetric systems giving rise to zero spontaneous mean

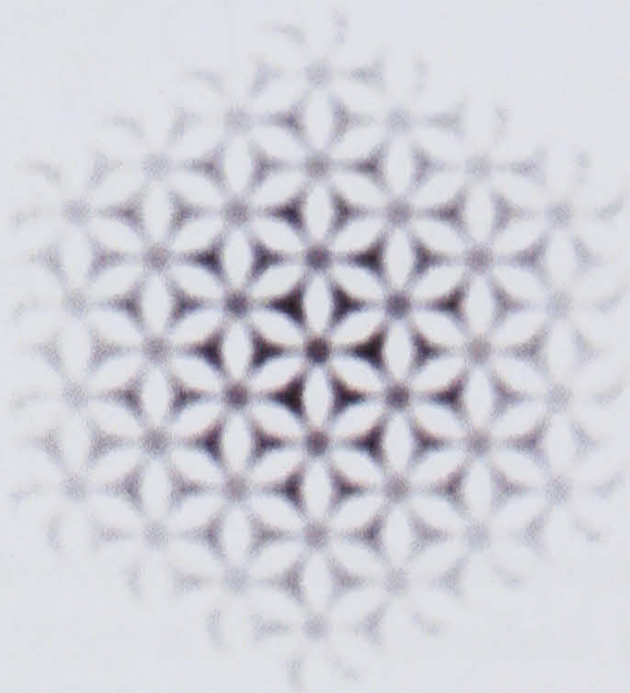


(a)

(b)



(c)



(d)

Figure 4.6: Views of the gyroid mesophase. 4.6(a) 2D slice from a small (16^3) simulation of a gyroid mesophase. Contours show composition, varying from pure oil to pure water; arrows show the surfactant orientation. It can be seen clearly that the surfactant sits at the interface, with the head groups pointing towards the water component. 4.6(b) Structure of the two labyrinths enclosed by a gyroid minimal surface, showing the characteristic triple junctions. 4.6(c) Channels running in the (100) direction of a gyroid surface: note how adjacent channels rotate in opposite senses. 4.6(d) Parallel-projection volume rendering of a gyroid, looking in the (111) direction to show the distinctive "wagon-wheel" appearance.

curvature, the gyroid was the most stable structure. Their argument was as follows: suppose that the system equilibrates in some given morphology, with lattice constant a, surface area A per unit cell, mean curvature $H(\mathbf{r})$, and Gaussian curvature $K(\mathbf{r})$. Including a surface energy σ per unit area with the Helfrich Hamiltonian from equation (4.3), the free energy of a unit cell of the system is

$$F = \sigma A + \int \left(2\kappa H^2 + \bar{\kappa} K\right) \mathrm{d}A. \tag{4.7}$$

The free energy density is therefore

$$f = \frac{(\sigma A^*)}{a} + \frac{1}{a^3}\left(2\kappa \int H^2 \mathrm{d}A + \bar{\kappa} \int K \mathrm{d}A\right) = \frac{(\sigma A^*)}{a} + \frac{1}{a^3} E_c, \tag{4.8}$$

where E_c is the curvature energy, and A^* is the dimensionless scaled area A/a^2, so that the terms in parentheses are independent of unit cell size a, for fixed morphology. Minimizing f with respect to unit cell size gives

$$a_{\min} = \left(\frac{3E_c}{|\sigma|\, A^*}\right)^{\frac{1}{2}} \tag{4.9}$$

$$f(a_{\min}) = -\frac{4}{27}\left(\frac{(|\sigma|\, A^*)^3}{E_c}\right)^{\frac{1}{2}}. \tag{4.10}$$

For a minimal surface, $H = 0$ everywhere, so the first term in E_c is zero. The second term may be simplified through the Gauss-Bonnet theorem[176], which states that

$$\int K \mathrm{d}A = 2\pi\chi \tag{4.11}$$

where χ is a constant known as the Euler characteristic, which depends only on the topology of the surface. The Euler characteristic will be described in more detail later on; for now it is sufficient to note that it takes a value of -8 per unit cell for a gyroid. The Gauss-Bonnet theorem simplifies the equilibrium free energy density to

$$f_{\min}|_{H=0} = -\frac{4}{27}\left(\frac{|\sigma|^3}{|\bar{\kappa}|}\right)^{\frac{1}{2}} \Gamma \tag{4.12}$$

where

$$\Gamma \doteq \left(\frac{A^{*3}}{2\pi\,|\chi|}\right)^{\frac{1}{2}} = \left(\frac{A}{2\pi a^6\,|\chi|}\right)^{\frac{1}{2}} \tag{4.13}$$

is called the topological index or homogeneity index[177, 178], and is independent of both lattice constant and choice of unit cell. Schwarz and Gompper calculated the value of Γ for several TPMS morphologies and showed that the gyroid had the highest value of Γ, giving it the lowest free energy of the TPMS phases. This perhaps explains why the gyroid is the more commonly experimentally observed TPMS phase; however, their analysis showed that a lamellar structure had a lower free energy still, giving a metastable gyroid. Moreover, as they pointed out, free-energy minimization with respect to lattice constant a and morphology are not independent, as assumed in the derivation above.

4.3 Dynamical simulations

In their 1995 review[138], Seddon and Templer claimed that "There is still no accurate free energy model for the cubic phases, which can correctly predict the phase sequence and relative stability". The situation is no longer quite as dire: the Helfrich membrane model[175] of Schwarz and Gompper shows reasonable agreement with the experimentally determined phase diagram for the cubic phases of a mixture of water, lauric acid, and the amphiphile dilauroyl phosphatidylcholine; their Ginzburg-Landau model yields further interesting results regarding phase stability[160].

However, despite the significant progress made in free energy models, which through analysis and Monte Carlo simulation give a good understanding of equilibrium properties and phase stability, comparatively little is known about the dynamics of the cubic phases, or indeed mesophases in general: free-energy functionals will not give information about the dynamics of a system far from equilibrium.

The limitation to equilibrium states is a severe one, for several reasons. Firstly, real-world systems may contain boundaries or imperfections, which prevent "perfect" equilibrium phases from forming. Secondly, rheological or otherwise dynamical properties of a mesophase are non-equilibrium by definition, and so cannot be captured by such treatments. Many of the mechanical properties of metals and electrical properties of semiconductors are determined primarily by the nature of defects and impurities, so an understanding of the non-equilibrium behaviour is essential to investigations of material properties.

Mesoscale techniques such as DPD or lattice Boltzmann, since they can handle kinetic descriptions, offer a non-equilibrium alternative to the free-energy descriptions. Groot and Madden[179] performed DPD simulations of diblock copolymer melts, which allowed reproduction of several well-known morphologies, and also showed the dynamical pathways through which they were reached. In particular, they showed the existence of a "gyroid-like" phase with many triple junctions. However, the phase was not exactly of cubic symmetry, possibly because of finite-size effects and frustration.

Using an early version of the LB3D code, Nekovee and Coveney[180] showed that it was possible to use the lattice Boltzmann model of Chen, Boghosian, and Coveney[68] to simulate the assembly dynamics of lamellar and bicontinuous phases of binary water-surfactant systems; indeed, they showed the existence of a P-surface cubic phase of the model. Then, while investigating the effect of the presence of surfactant on spinodal decomposition[181], González and Coveney discovered[182, 183] the existence of a gyroid cubic phase. Importantly, use of the lattice Boltzmann method allowed modelling of the dynamics of the self-assembly of the gyroid.

The lattice Boltzmann simulations of the gyroid phase demonstrated several important features. Firstly, it had been suggested[184] that simulation of cubic phases might require a model with more complicated amphiphiles than symmetric dimers, and also that long-range interactions might be required. The lattice Boltzmann simulations, which used symmetric amphiphiles with short-range interactions, refuted this. Secondly, analysis of the X-ray structure factor of the simulation data showed the existence of oscillating modes at long times; visualization of the simulation output strongly suggested that these were due to Marangoni effects.

Marangoni flows[185, 186] are macroscopic fluid flows generated by surface tension gradients. If a glass of wine is tilted and then set down on a level surface, "tears" of wine can often be seen forming and running down from the top of the

thin film of wine adhering to the side of the glass. This happens due to Marangoni flow: the alcohol in the wine evaporates near the top of the film, causing a composition gradient and therefore a surface tension gradient, which draws liquid upwards until it can no longer be supported, and rolls down as a tear[187]. González and Coveney demonstrated the presence of surfactant gradients in the gyroid phase, and the oscillations were attributed to the resulting Marangoni flows.

Free-energy surfactant mesophase models typically restrict the amphiphile to occupy the oil-water interface; such models therefore cannot describe amphiphile gradients, and cannot reproduce the effects of Marangoni flows.

Theissen *et al* [188] describe a lattice Boltzmann model of amphiphilic systems which is based around a Ginzburg-Landau free energy functional[189, 160]; however, it should be noted that this model lacks an explicit surfactant density, making the assumption that the surfactant is all adsorbed onto the oil-water interface. It is not expected, therefore, that this model would reproduce Marangoni effects at the interface. Lamura *et al* [190] describe a Ginzburg-Landau model with explicit surfactant density, but their model in turn lacks explicit surfactant orientation. The model used in LB3D has both explicit density and orientation for surfactant particles. Moreover, this model exhibits a surprising degree of stability: simulations have been run with LB2D and LB3D up to order 10^6 timesteps[191], without problems. Other three-dimensional LB models have had difficulty reaching these timescales[96], although attempts have been made[192] to correct these problems.

The gyroid self-assembly simulations used initial conditions of a randomized mixture of oil, water, and surfactant, with the oil and water in 1:1 ratio. Phase separation would occur rapidly, within the first thousand timesteps or so. After phase separation, the system would form a "molten gyroid" phase, consisting of many oil or water rods surrounded by surfactant; these rods would join up to form triple junctions, giving rise to a gyroid morphology. Towards the end of these simulations, after around 30000 timesteps, the Marangoni oscillations were observed.

These simulations were originally performed to investigate the dynamics of surfactant-limited phase separation; discovery of the gyroid phase was an interesting side effect. Because of this, the simulations of González and Coveney were not optimally suited to examining the gyroid phase, in several ways.

Firstly, they were limited in size: most of the simulation work was done with 64^3 systems. These are perfectly adequate for phase-separation studies; however, it was observed that while at early times there might be several different gyroid regions with different orientations, at long times these would join up to form a single gyroid grain spanning the entire simulation grid, a so-called "perfect gyroid". In the real world, cubic mesophases are far from perfect: despite careful experimental procedures[193] artificially created gyroid materials may not consist of a perfect gyroid repeating throughout space, but rather of many gyroid grains with different orientations. In addition, there may exist dislocations and defects within these grains, analogous to the defects found in ordinary solid crystals[194, 195]. Any investigation of the gyroid domains or the nature of their interactions would require a simulation of sufficient physical size to accommodate several domains for the majority of the simulation.

Secondly, they were limited in time: the timescales probed (maximum of 35000 timesteps) offered few hints as to how transient the effects observed were. Phase separation was observed to happen extremely quickly, over a thousand timesteps or so, and formation of gyroid morphology on a slower but comparable timescale. The behaviour of gyroid defects, on the other hand, takes place on a much slower scale.

Parameter	Set 8	Set 9
Initial oil density n^{r}	0.7	0.7
Initial water density n^{b}	0.7	0.7
Initial surfactant density n^{b}	0.9	0.6
Oil-water coupling g_{cc}	0.08	0.08
Oil-surfactant coupling g_{cs}	-0.006	-0.006
Surfactant-surfactant coupling g_{ss}	-0.003	-0.0045

Table 4.1: Parameter sets used in TeraGyroid simulations

at least tens of thousands of timesteps, and these timescales were not probed.

With these limitations in mind, a new set of simulations was performed specifically to look at the nature of defect dynamics in the gyroid liquid crystal phase.

4.4 The TeraGyroid simulations

It was clear that larger simulations were required; how much larger was not so clear, since defects had not been observed in the unambiguous absence of finite-size effects.

Defect dynamics simulations were performed as part of a larger project[196] to construct a transatlantic distributed simulation Grid. As part of this project, a significant amount of computing power became available for use. One of the aims of this project was to investigate the use of computational steering[71] in soft condensed matter simulations; finding and examining defect dynamics in a lattice Boltzmann model is an ideal application of steering.

Hence, some of the parameter sets found by González and Coveney to produce gyroid phases were chosen, and simulations of these parameter sets were initialized. The intention was to run a set of simulations of different sizes, and visualize them as they ran: if any of the smaller systems became uninteresting due to formation of "perfect" gyroids, then they could be terminated, and their resources freed used to tackle larger problems (see appendix B).

The simulations of most interest used the parameter sets González [183] called numbers 8 and 9, which are shown in table 4.1.

Simulations were run at sizes of, amongst others, 64^3, 128^3, and 256^3, for durations listed in table 4.2. If a stable gyroid state was reached, then a simulation was terminated; the systems which did not reach this state ran for as long as possible under the resource constraints at the time.

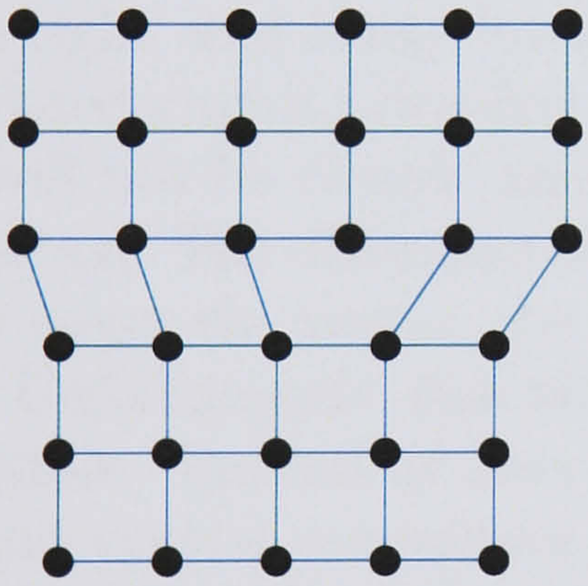
During each simulation, the order parameter field $\phi(\mathbf{r}) = \rho^{\mathrm{r}}(\mathbf{r}) - \rho^{\mathrm{b}}(\mathbf{r})$ was stored to disk, as well as the surfactant density field $\rho^{\mathrm{s}}(\mathbf{r})$. This was performed every Δt timesteps, for values of Δt also shown in table 4.2. The output rate was chosen according to several constraints: it had to be sufficiently rapid to allow observation of the gyroid dynamics, but limits were imposed by local disk quotas, which would often shift due to data from other concurrent jobs.

4.5 Properties of simulation output data

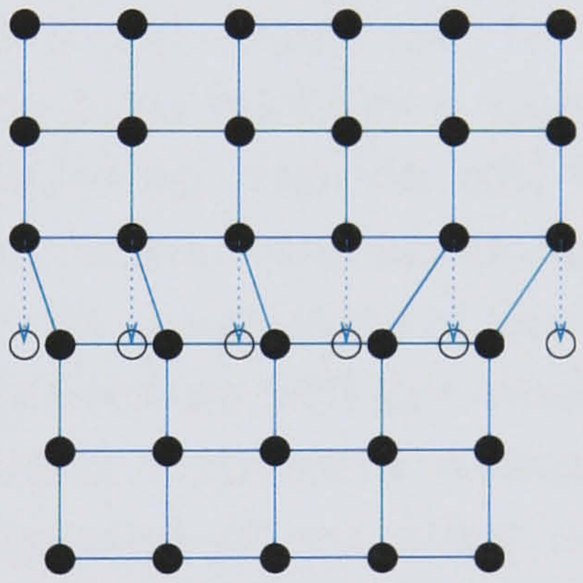
Consider a dislocation defect in a simple crystal structure (figure 4.7(a)). In the plane shown, there are two Bravais lattice vectors which, away from the defect region, join each atom to its neighbours. The defect region can be easily identified,

	System	$t_{\max}$	Output rate Δt	Final state
Set 8:	16^3	5000	50	Perfect gyroid
	32^3	50000	500	Perfect gyroid
	64^3	200000	100	Gyroid with dislocation pair
	128^3	384000	250	Single gyroid domain with point defects
	256^3	57500	100	Multiple gyroid domains
Set 9:	16^3	5000	50	Perfect gyroid
	32^3	100000	500	Perfect gyroid
	64^3	250000	500	Skewed gyroid
	128^3	999900	100	Gyroid with dislocation pair
	256^3	146500	500	Multiple gyroid domains

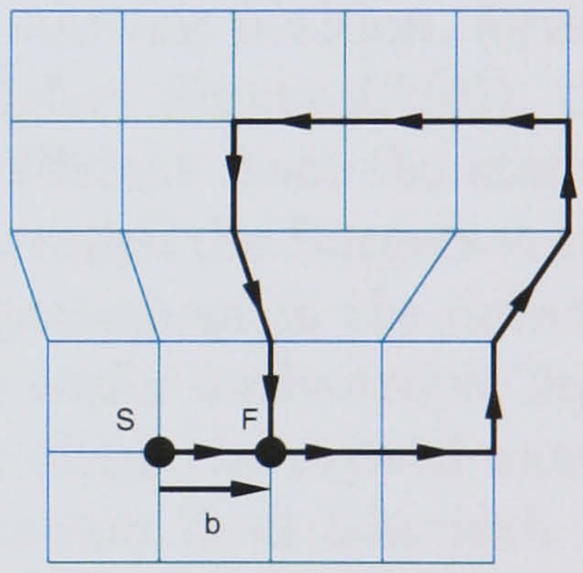
Table 4.2: Simulation durations and end states.



(a) Dislocation in a simple crystal plane, showing lattice vectors

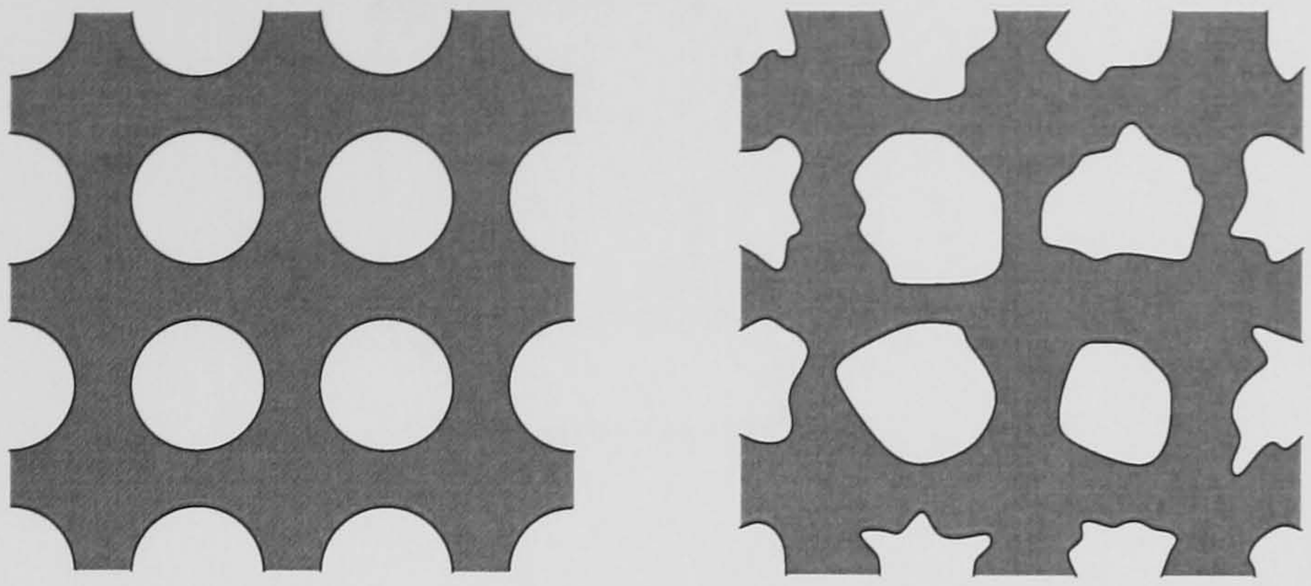


(b) Translation along the lattice vectors in the region of the defect does not result in the location of another atom



(c) Burgers circuit around the dislocation: a path which would be closed in a perfect lattice is not closed when it encloses a dislocation. The difference between the start position S and finish position F is called the Burgers vector **b**.

Figure 4.7: Dislocations in a simple crystal.



(a) Plane through a regular liquid crystal structure.

(b) The same liquid crystal, with surface perturbations.

Figure 4.8: Two topologically identical liquid crystal structures.

because translation along a lattice vector from one of the adjoining atoms does not result in the location of another atom.

Therefore, in a crystal composed of atoms or molecules whose positions are known, a defect is straightforward to locate. Furthermore, if the links between atoms are known or can be found, then the Burgers vector, which describes the direction and magnitude of lattice displacement in a defect, can also be determined[195]. For example, if there were no defects in the structure in figure 4.7, then it would have a square lattice, and moving three lattice links to the left, three upwards, three to the right, and three downwards would return you to the starting position, forming a closed circuit. However, if this path encloses a single defect (figure 4.7(c)), then it will not be closed, and the finishing position will be different from the starting position. The difference between the start and endpoints is called the Burgers vector: the larger the vector, the larger the magnitude of the displacement in the defect.

Unfortunately, this kind of procedure will not work so easily for lyotropic liquid crystals. Instead of discrete atomic structures repeating along the crystal axes, a liquid crystal mesophase is composed of repeating continuum fluid labyrinth elements. Because the mesophase labyrinth is a continuum, it is impossible to define a single point "position" in the same way that one can for an atom. Moreover, the labyrinth itself can fluctuate and ripple without changing its topology, and therefore recognizably retaining the same overall structure (figure 4.8).

Hence, direct automatic location of defect regions is a very difficult problem[197]. Nonetheless, there are several properties of the fluid which can be calculated and used to provide information about the nature of the mesophase.

4.5.1 Polygonal approximation of the interface

In a continuum system, the interface between the oil and water phases can be defined as the implicit surface $\phi(\mathbf{r}) = \rho^{\mathrm{r}}(\mathbf{r}) - \rho^{\mathrm{b}}(\mathbf{r}) = 0$ for $\mathbf{r} \in \mathbb{R}^3$. In contrast, the simulation output data consists of the values of ϕ at discrete sites $\mathbf{r}_i$ on the lattice: it is not quite so straightforward to define an interfacial surface because in general, while $\phi(\mathbf{r}_i)$ can be positive or negative, corresponding to the lattice site containing a majority of red or blue particles respectively, it is almost never exactly zero.

However, a continuum order parameter field can still be estimated from the

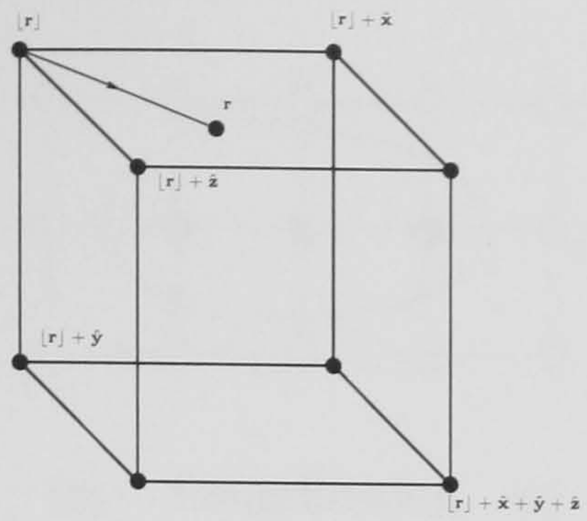


Figure 4.9: Any point $\mathbf{r}$ lies inside the unit cube with a corner at $\lfloor \mathbf{r} \rfloor$.

discrete simulation output, by linear interpolation. For any point $\mathbf{r} = (x, y, z)$ inside the simulated region but not necessarily lying on a lattice site, define $\lfloor \mathbf{r} \rfloor = (\lfloor x \rfloor, \lfloor y \rfloor, \lfloor z \rfloor)$, where $\lfloor x \rfloor$ is the integer part of x. The point $\mathbf{r} = \lfloor \mathbf{r} \rfloor + \boldsymbol{\lambda}$ then lies inside the unit cube whose low-(x, y, z) corner is $\lfloor \mathbf{r} \rfloor$, as shown in figure 4.9. The trilinearly interpolated value $\phi^{\text{interp}}(\mathbf{r})$ is then[198, 199, 200]:

$$\begin{aligned}
\phi^{\text{interp}}(\mathbf{r}) \doteq\ & (1-\lambda_x) \cdot (1-\lambda_y) \cdot (1-\lambda_z) \cdot \phi_{000} \\
+\ & \lambda_x \cdot (1-\lambda_y) \cdot (1-\lambda_z) \cdot \phi_{100} \\
+\ & (1-\lambda_x) \cdot \lambda_y \cdot (1-\lambda_z) \cdot \phi_{010} \\
+\ & \lambda_x \cdot \lambda_y \cdot (1-\lambda_z) \cdot \phi_{110} \\
+\ & (1-\lambda_x) \cdot (1-\lambda_y) \cdot \lambda_z \cdot \phi_{001} \\
+\ & \lambda_x \cdot (1-\lambda_y) \cdot \lambda_z \cdot \phi_{101} \\
+\ & (1-\lambda_x) \cdot \lambda_y \cdot \lambda_z \cdot \phi_{011} \\
+\ & \lambda_x \cdot \lambda_y \cdot \lambda_z \cdot \phi_{111},
\end{aligned} \tag{4.14}$$

where $\phi_{ijk} = \phi(\mathbf{r} + i\hat{\mathbf{x}} + j\hat{\mathbf{y}} + k\hat{\mathbf{z}})$. The interpolated function is single-valued and continuous across the simulated region, and can be regarded as the simulated approximation to the order parameter field; the isosurface $\phi^{\text{interp}}(\mathbf{r}) = 0$ is then well-defined.

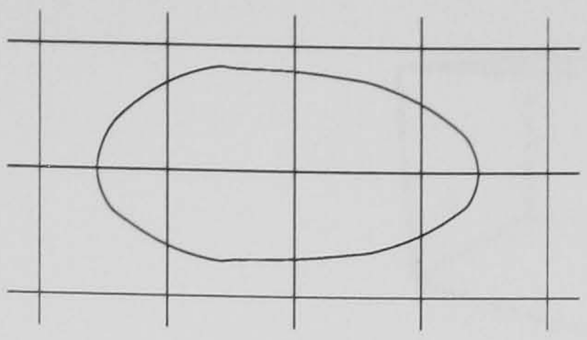
Generation of a polygon mesh approximation to this surface can be achieved through several means. The most popular method is known as Marching Cubes: this technique was developed by Lorensen and Cline[201] .

In Marching Cubes, each point $\mathbf{r}$ on the lattice is assigned a value of 1 if $\phi(\mathbf{r})$ is "negative", or unambiguously less than zero, and 0 if $\phi(\mathbf{r})$ is "positive", *i.e.* greater than zero or within floating-point error of zero. This classification is illustrated in figure 4.10(b).

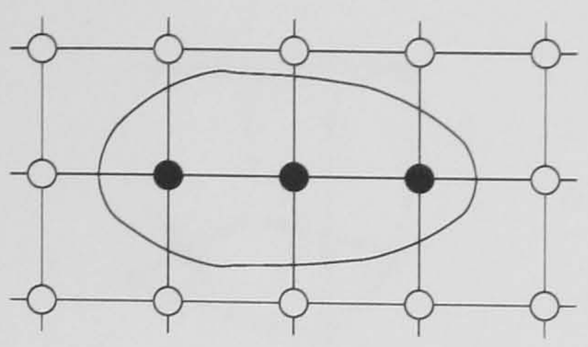
If a positive site is adjacent to a negative site, then $\phi(\mathbf{r})$ changes sign somewhere along the link between the two sites, and therefore the link between the sites intersects the contour surface; the position of these intersections can be found easily through interpolation.

The volume occupied by the lattice can be divided into cubes, each bounded by eight lattice sites. Since each site is assigned a state of 1 or 0 depending on whether it is above or below the contour surface, there are $2^8 = 256$ possible states for each cube. The symmetry operations of reflection, rotation, and Boolean negation reduce this to 15 cases, as shown in figure 4.11.

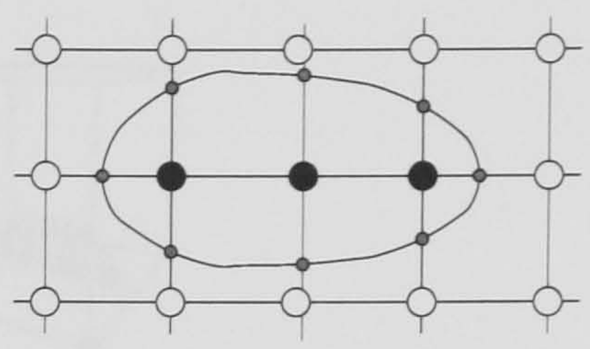
The Marching Cubes algorithm proceeds by considering each cube in turn, calculating a number between 0 and 255 representing its state, and then using this number to find a set of polygons approximating the interface shape inside the cube. The polygon sets can be stored in a lookup table, making the algorithm extremely



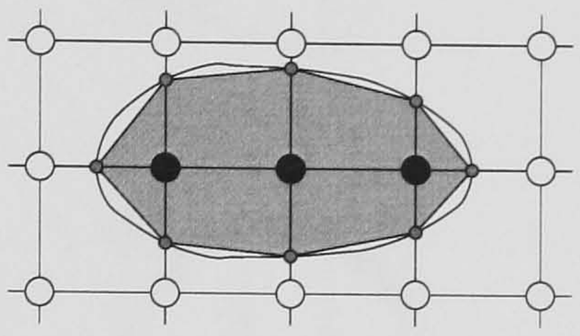
(a) Contour of an interpolated function.



(b) Classification of vertices into points above and below the contour.



(c) Calculation of intersections.



(d) Polygonal approximation to isosurface.

Figure 4.10: Calculating the contours of a function interpolated from points on a lattice.

fast.

Unfortunately, the algorithm is not guaranteed to produce a surface with the same topology as the implicit interpolated surface $\phi^{\text{interp}} = 0$. In fact, it is possible for Marching Cubes to produce "non-manifold" surfaces containing holes: this arises because some of the 256 possible states of a cube have more than one way of being tiled with polygons, and the tilings are topologically different.

Several approaches have been suggested to correct this problem. One such technique is to tile the space with tetrahedra rather than cubes[202, 203], which produces a topologically consistent surface without holes, but at the expense of speed, memory, and geometrical accuracy[204]. Another approach is the recursive Dividing Cubes algorithm[205], which has similar performance penalties, particularly for large datasets.

Chernyaev[206] observed that it was possible to extend the Marching Cubes approach to produce polygon surface meshes which are not only topologically consistent (that is, do not have any holes), but also topologically correct (that is, homeomorphic to the surface $\phi^{\text{interp}} = 0$).

There are two sorts of ambiguity in Marching Cubes: *face ambiguities* and *internal ambiguities*. Face ambiguities occur when, on a given face, two diagonally opposite vertices are above the surface, and the other two are below it: the ambiguity is over whether the surface should join the two vertices above the surface, or whether it should separate them, as shown in figure 4.12. Internal ambiguities arise in ascertaining whether or not two vertices which are on different faces of the cube are joined by a tunnel running through the cube, as seen in figure 4.13. These cases can be resolved by examining the value of the interpolated function on or inside the cube, and using this information to index extra lookup tables containing the tilings for the ambiguous cases.

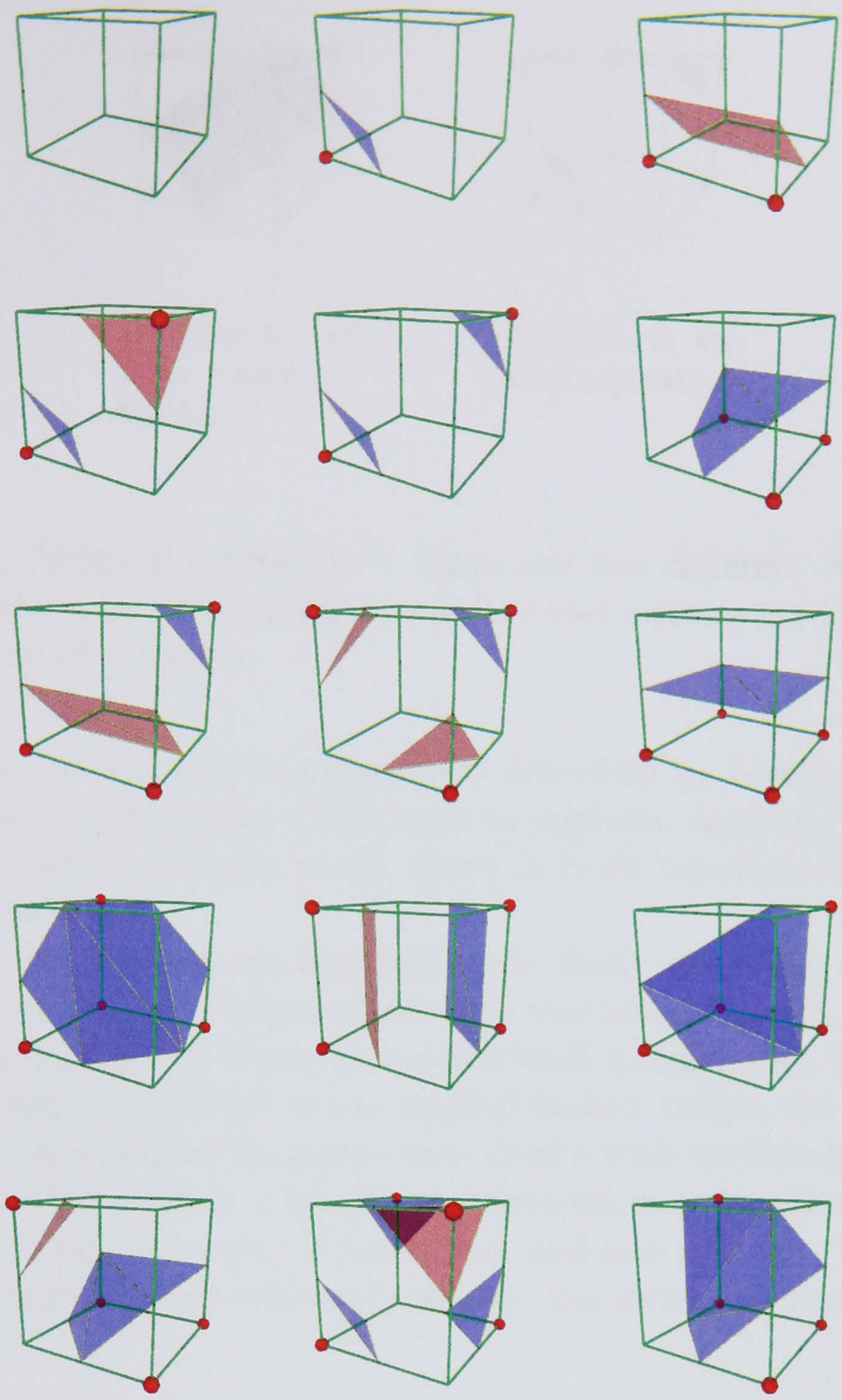
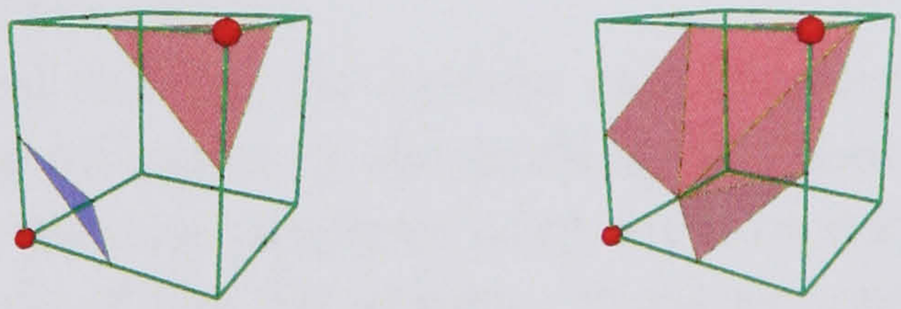


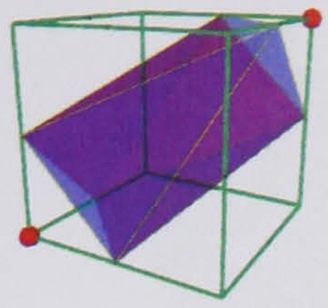
Figure 4.11: The fifteen classical Marching Cubes configurations



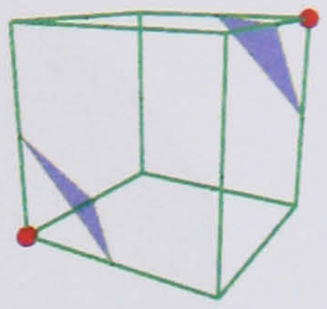
(a) Case 1: vertices separate.

(b) Case 2: vertices joined.

Figure 4.12: A "face ambiguity" in Marching Cubes: while the state of the vertices remains unchanged, there are two different ways of drawing a surface, depending on whether or not two vertices on the same face are joined. Vertices with red blobs are on one side of the surface; unmarked vertices are on the other.



(a) Case 1: vertices joined internally.



(b) Case 2: vertices separate.

Figure 4.13: An "internal ambiguity": there are two different ways of drawing a surface through the cube, depending on whether two vertices are joined by a passage through the centre of the cube.

An implementation of this technique was described by Lewiner *et al* [204], who used lookup tables to determine which tests to perform, resulting in a very efficient algorithm to produce a triangle mesh which is both topologically consistent and topologically correct.

The source code for this implementation is freely available, and the program used for analysis of the gyroid simulation data was based upon it. However, several changes were made (some of which have since been merged with the original code): in addition to fixing some errors in the original lookup tables, the tiling tables were recalculated to ensure that all polygons were drawn with vertices in clockwise order, looking from inside the surface. This allowed the surface to be coloured differently on each side when the polygon mesh is visualized, and also permits "backface culling", the removal of polygons which will not be seen by the viewer, to improve visualization performance.

4.5.2 Fluid properties at the interface

By applying this algorithm to the simulation output datasets, a polygonal approximation to the interface between the two fluid components can be found. This is useful not only for direct visualization of the system, but also for calculating many interesting properties of the system.

Firstly, the interfacial area of the system can be calculated, simply by adding up the areas of all of the triangles in the surface. Without knowing the location of the interface, any space-varying property $\alpha(\mathbf{r})$, such as surfactant density, can only be averaged over the bulk of the 3D system. However, once the interface has been polygonized, the variation of α over the interface can now be found, by interpolating α to the surface vertices. A naïve procedure for calculating the average value of α over the interface would be to calculate its value at each vertex, and take the average of these values. However, this method gives equal weight to each vertex. Regions of high detail and curvature may be tiled by many small polygons instead of few large ones, with the result that there will be more vertices in such regions: the naïve method is therefore biased towards such regions.

A better procedure is to calculate an area-weighted average. Consider a single triangle Δ_i, area A_i, vertices $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i$. The value of α averaged over the triangle is

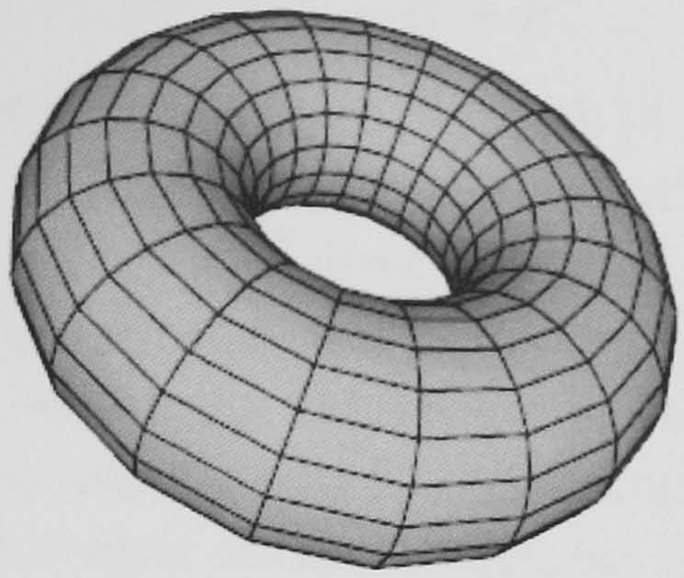


Figure 4.14: A closed surface, tiled with polygons.

$$\alpha_i = \frac{1}{A_i}\int_{\Delta_i} \alpha(\mathbf{r})\ \mathrm{d}^2 A_i \simeq \frac{1}{3}\left[\alpha(\mathbf{a}_i) + \alpha(\mathbf{b}_i) + \alpha(\mathbf{c}_i)\right]. \tag{4.15}$$

This relationship is exact if α varies linearly over the surface of the triangle. Integrating α over the entire closed polygon mesh M is equivalent to summing the values integrated over each individual triangle:

$$\oint_M \alpha(\mathbf{r})\ \mathrm{d}^2 A = \sum_i \int_{\Delta_i} \alpha(\mathbf{r})\ \mathrm{d}^2 A_i \simeq \sum_i \alpha_i A_i \tag{4.16}$$

Hence the average of α over the surface is

$$\langle\alpha\rangle_M = \frac{\sum_i \alpha_i A_i}{\sum_i A_i}. \tag{4.17}$$

4.5.3 Interfacial curvature

Any point $\mathbf{r}$ on the surface $\phi(\mathbf{r}) = 0$ has a unit normal $\hat{\mathbf{n}}$, defined as

$$\hat{\mathbf{n}}(\mathbf{r}) \doteq \frac{(\nabla\phi)}{|\nabla\phi|}. \tag{4.18}$$

The Shape Operator $\mathbb{S}$ is a rank-2 tensor defined as

$$\mathbb{S} \doteq -\nabla\hat{\mathbf{n}} \tag{4.19}$$

The Gaussian and mean curvatures can then be found[176] from the determinant and half the trace of the shape operator, respectively. More usefully, the surface-averaged value of the squared mean curvature, $\langle H^2\rangle_{\mathrm{surf}}$ can also be calculated: this quantity is physically interesting because it appears as a term in the Helfrich Hamiltonian.

4.5.4 Interfacial Euler characteristic

Consider a two-dimensional closed surface tiled with a polygon mesh, such as the torus in figure 4.14. Each polygon is joined to its neighbours by a straight edge; the edges meet at point vertices. Suppose that the surface has V vertices, E edges, and F polygon faces; the Euler characteristic or Euler number of the surface is defined as

$$\chi^{\text{surf}} \doteq V - E + F. \tag{4.20}$$

It turns out that, whichever way the torus in figure 4.14 is covered in non-overlapping polygons, the Euler number will always be zero; similarly, the Euler number of a polygonal surface covering a sphere will always be two. This generalizes to the surprising and very powerful result of elementary topology[207, 208, 209] that two topologically equivalent surfaces, which can be deformed into one another by stretching but not cutting, have the same value of the Euler characteristic. More precisely[207], if S_1 and S_2 are compact, connected surfaces without boundary, then S_1 is topologically equivalent to S_2 if and only if $\chi(S_1) = \chi(S_2)$, and either both are orientable or both are non-orientable. A surface is non-orientable if it contains Möbius bands: all of the surfaces under consideration here are orientable, so determination of topological equivalence reduces to comparison of the Euler characteristic.

The Euler number is additive: a system with n surfaces S_i each with Euler number χ_i has total Euler number $\sum_i \chi_i$.

The surface meshes produced by the Marching Cubes algorithm are composed entirely of triangles: this property simplifies calculation of the Euler characteristic. Each triangle on the mesh has three edges; each edge on the mesh joins exactly two triangles. Hence, the number of edges is 3/2 times the number of triangles:

$$\chi^{\text{tri}} = V - E + F = V - \frac{1}{2}F. \tag{4.21}$$

The Euler number of a surface is directly related to another topological quantity called the genus g, which is often regarded as simply the number of holes in the surface. For an orientable surface[207]:

$$\chi = 2(1 - g) \tag{4.22}$$

The Euler number is an extremely useful concept when trying to understand the output of mesoscale simulations. Imagine a lattice Boltzmann simulation of a spherical-droplet phase, where the droplets are defined as regions for which the order parameter $\phi(\mathbf{r}) < 0$. Since the Euler number of a sphere is 2, and since it is additive, the number of droplets in the system is simply half the Euler number of the $\phi = 0$ surface. Moreover, the Euler number, being a topological invariant, is insensitive to fluctuations in the droplet shape or size, provided that each droplet remains simply-connected.

Recently, the Euler number was used by Hołyst and Oswald[210] to characterise fluctuating wormhole-like passages[211] appearing in Monte Carlo simulations of lamellar phases; similar wormholes have been observed in a lattice gas dynamical model[212]. In a system with periodic boundary conditions, a membrane spanning both the x and y directions is topologically equivalent to the surface of a torus, and therefore has Euler number zero; thus, a lamellar stack of membranes will also have Euler number zero. However, if a wormhole forms between two lamellae, the Euler number will suddenly jump to -2, hence the Euler number is $-2n$ for n such wormholes.

The Euler number has also been used in examinations of TPMS surfactant morphologies[160, 152, 157, 150] and experimental MRI data[213], as well as being an important parameter in advanced isosurfacing algorithms[214] which attempt to reduce the number of polygons in the mesh without changing its topology.

The concept of Euler number can be generalized to higher-dimensional objects: for a three-dimensional volume composed of C polyhedra. in turn composed of F faces, meeting in E edges, and V vertices, the three-dimensional Euler number is

$$\chi^{\text{vol}} \doteq V - E + F - C. \tag{4.23}$$

In the most general case, for an N-dimensional object K composed of n_0 zero-dimensional points, n_1 one-dimensional edges, and, in general, n_i i-dimensional objects, "the" Euler number can be written

$$\chi \doteq \sum_{j=0} (-1)^j n_j. \tag{4.24}$$

Since they have different values, it is important to be clear about which Euler characteristic is being used: the 3D number[214, 215, 216] or the 2D one[152, 160, 150, 210], although seemingly few authors[213] make it completely explicit which version they use. It can be assumed below that the Euler characteristic is the 2D surface version χ^{surf} (equation 4.20), unless explicitly stated otherwise.

4.5.5 The structure function

A commonly examined property of mesoscale systems is the *structure function*, which is defined below for a three-dimensional system in a manner analogous to that in which a two-dimensional structure function was defined in chapter 2. Define the pair correlation function in terms of the the order parameter $\phi(\mathbf{r})$, as

$$g(\mathbf{r}) \doteq \int \phi(\mathbf{r}')\phi(\mathbf{r}+\mathbf{r}')\ \mathrm{d}\mathbf{r}'. \tag{4.25}$$

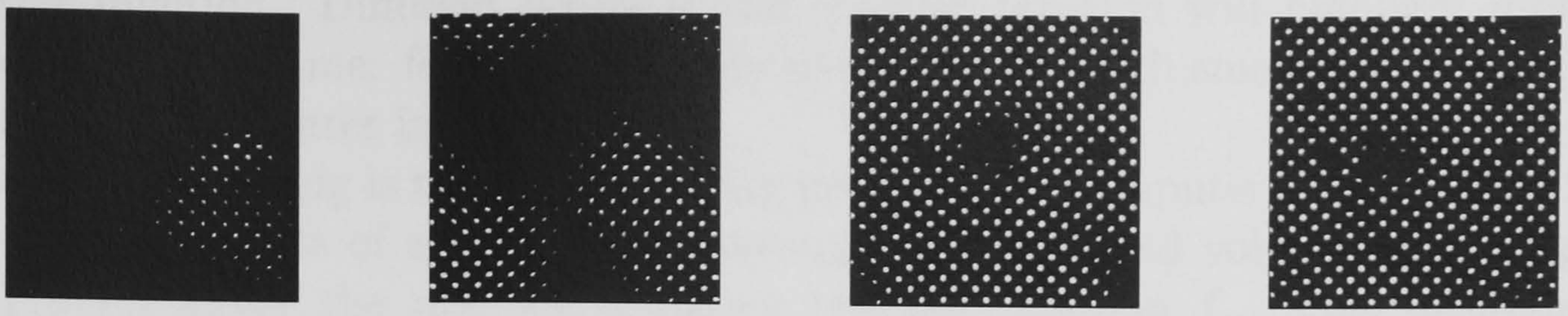
The Fourier transform of this function turns out to be $|\tilde{\phi}(\mathbf{k})|^2$, the squared magnitude of the Fourier transform of the order parameter. The spherical average of this gives the structure function:

$$S(k) \doteq \left\langle \left| \int \phi(\mathbf{r}) \exp\left(i\mathbf{k}\cdot\mathbf{r}\right)\ \mathrm{d}\mathbf{r} \right|^2 \right\rangle_{|\mathbf{k}'|=k}. \tag{4.26}$$

This is very closely related to the "structure factor" measured in small angle X-ray scattering (SAXS) experiments, which are commonly used in experimental examination of mesophases[162, 161, 217, 218], and has been an important tool in the analysis of many mesoscale simulations[82, 81, 182, 86]. To measure the dominant length scale of a system, the reciprocal space first moment k_1 of $S(k)$ is usually calculated:

$$k_1 \doteq \left(\int kS(k)\ \mathrm{d}k \right) / \left(\int S(k)\ \mathrm{d}k \right). \tag{4.27}$$

The corresponding length scale is then $L_1 \doteq 2\pi/k_1$. Unfortunately, $S(k)$ only measures averaged bulk properties of the system, and does not supply enough information to locate defects in space[197].



(a) Timestep 10^5

(b) Timestep 3×10^5

(c) Timestep 5×10^5

(d) Timestep 7×10^5

Figure 4.15: Thresholded slices of the 128^3 parameter set 9 simulation at various timesteps. White pixels indicate regions where a channel of one particular component runs through the entire slice without interruption.

4.5.6 Thresholding and slicing

A number of techniques were used by Harting *et al* [197] in an attempt to spatially localize defects in the gyroid structure. These techniques were based around taking slices of each order parameter dataset and turning each slice into a 2D image, with a pixel set to black if any of the corresponding lattice sites have an order parameter less than some threshold, and leaving it white otherwise. Typically, the algorithm worked best when the slice size was manually tuned to be roughly the same size as a unit cell of the gyroid.

This produced images showing the gyroid channels as a regular pattern of dots (figure 4.15). The pattern recognition algorithm of Chetverikov and Hanbury[219, 220] was then used to find regions where this regular pattern was distorted or interrupted. This technique could easily match point defects and line defects; however, it is still difficult to use the technique to locate domain walls. Moreover, the slicing of the lattice is an inherently anisotropic technique, and requires the assumption that the gyroid is roughly aligned with the simulation grid: an assumption which is not always safe to make, particularly at early times, and renders the technique vulnerable to the sort of erroneous perception of defect lines illustrated in figure 4.25, when defect-lke patterns appear which are in fact due to the gyroid's orientation with respect to the viewer, rather than due to the existence of an actual defect.

4.5.7 Volume rendering

A common method of visualizing the order parameter field is to polygonize a constant-ϕ surface as described in section 4.5.1, and then render the resulting polygons to the screen. However, complicated surface geometries can result in very high numbers of polygons being generated, making this a slow and expensive way of examining the output from large complex fluid simulations.

An alternative method is called *volume rendering*. Imagine the order parameter field from a simulation represented as a block of glass with sides in the same ratio as the simulation box, with each point inside the glass block dyed red if the value of ϕ at the corresponding point inside the simulation is less than zero, and left transparent otherwise. The result would be a three-dimensional representation of the oil-filled regions of the simulation. The mapping from values of $\phi(\mathbf{r})$ to the colour and transparency of the corresponding piece of the glass block is called the

transfer function. Different forms of the transfer function will highlight different aspects of the volume: for example, only dying regions with small values of $|\phi|$ will highlight the oil-water interface.

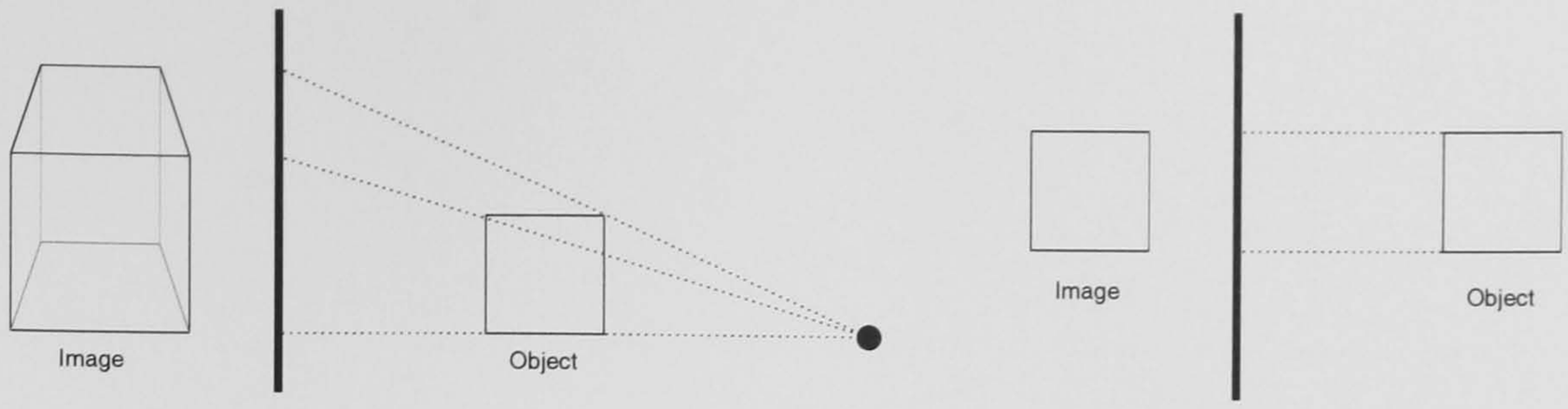
Volume rendering is the corresponding procedure in computer graphics[200, 221]. It typically consists of shooting rays through the simulated volume, and integrating $f(\phi(\mathbf{r}))$ along the ray, for a chosen transfer function f. This technique is well suited to lattice-based data, and variants of it can be implemented using the texture-mapping ability of modern graphics cards to achieve extremely high rendering speeds[222, 223].

Whether displaying the simulation data as polygonized surfaces or through volume rendering, it is important to choose an appropriate viewing transformation, or projection. In perspective projection (figure 4.16(a)), lines are drawn from a single point in space, through a point on an object, until they intersect with the projection plane, when the corresponding pixel is drawn on the viewer's screen. Rendering using this technique means that more distant objects are drawn as smaller, giving a good sense of depth. For example, the perspective projection polygonized surface in figure 4.25(a) shows the gyroid channels running away from the viewer. In contrast, the orthogonal or parallel projection of figure 4.16(b) uses lines drawn parallel to the projection plane, so that objects with the same size are drawn on-screen with the same size, regardless of their distance from the viewer. While this projection can feel very distorted and unintuitive to interact with, it is very good at showing up certain features which would otherwise be lost by the perspective viewing transformation: for example, the orthographic projection in figure 4.25(b), while it gives no clues as to how deep the gyroid channels are, clearly demonstrates the ordered and slightly skewed arrangement of the channels, which is much harder to see in the perspective projection. Another useful feature is that volume rendering with orthographic projection is closely analogous to transmission electron micrography, which fires a beam of electrons through a sample, and displays the orthographically-projected result.

Most commonly-available visualization toolkits, such as VTK[224], permit the transfer function to be a function only of the scalar field ϕ. However, Kniss *et al* [225] describe a volume rendering algorithm which permits the transfer function to depend on first and second derivatives of ϕ, to highlight regions according to their gradient; this technique has been implemented in the Simian[226] volume rendering code. While no accurate algorithm has yet been found for locating gyroid defects in space, it was found through trial-and-error that highlighting regions with a low value of $|\phi|$ and a simultaneously low value of $|\nabla\phi|$ would show up domain walls in gyroid structures in a way which can be picked out clearly by the human eye, if still marred by noise. A possible explanation for the effectiveness of this algorithm is that it picks out regions near the interface ($|\phi| \simeq 0$) which also have a low gradient ($|\nabla\phi| \simeq 0$) and therefore low curvature, which is more likely to occur in the flattened "blob" regions in domain walls shown in figure 4.27. Refinement of this procedure to determine its origin and extract quantitative results is an obvious area for future exploration.

4.6 Analysis of the simulation data

So far, both the techniques used to simulate assembly of the gyroid mesophase and the techniques used to analyse the resulting data have been presented. The following section presents the actual analysis of the generated data, and a discussion of the



(a) Perspective projection

(b) Orthographic or parallel projection

Figure 4.16: Two different projection techniques.

physics involved.

4.6.1 Initial phase separation

Recall from section 4.4 that the system is initialized to a very highly mixed and disordered state, often called a "quench" in the literature[94, 39, 86]. The system undergoes spinodal decomposition[72, 181, 191, 93, 94], and rapidly divides into interpenetrating regions of bulk water and bulk oil, separated by a layer of surfactant. If the system contained only water and oil, each component would eventually form a single large domain; however, this situation would leave insufficient surface area to allow all of the surfactant to sit at the oil-water interface. Therefore, the surfactant has the effect of limiting the domain growth[227].

Figure 4.17 shows the dominant length scale $L_1(t)$ for the first 5000 timesteps of three simulations using parameter set 8. This shows that the length scale rapidly increases as the oil and water components separate, but stays relatively constant after around 1000 timesteps. Figure 4.18 shows renderings of the oil-water interface at $\phi = 0$ over the first 500 timesteps, indicating that the rapid increase in structure factor corresponds to the formation of a mesh of interconnecting channels, all of roughly the same width. Visual inspection of the interface shows that no gyroid structure is present during the channel formation.

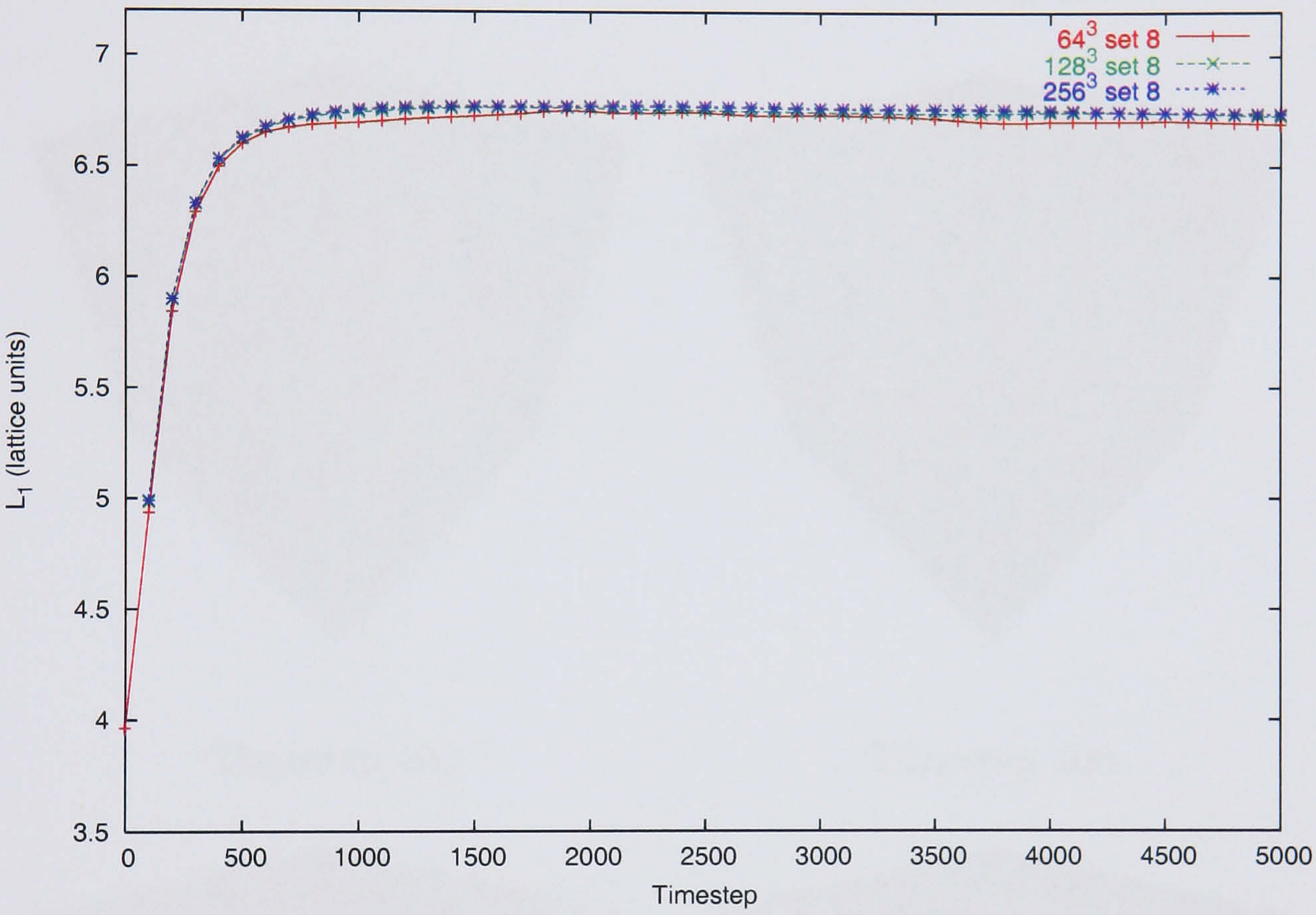


Figure 4.17: Typical length scale $L_1(t)$ as a function of time, for three simulations on parameter set 8.

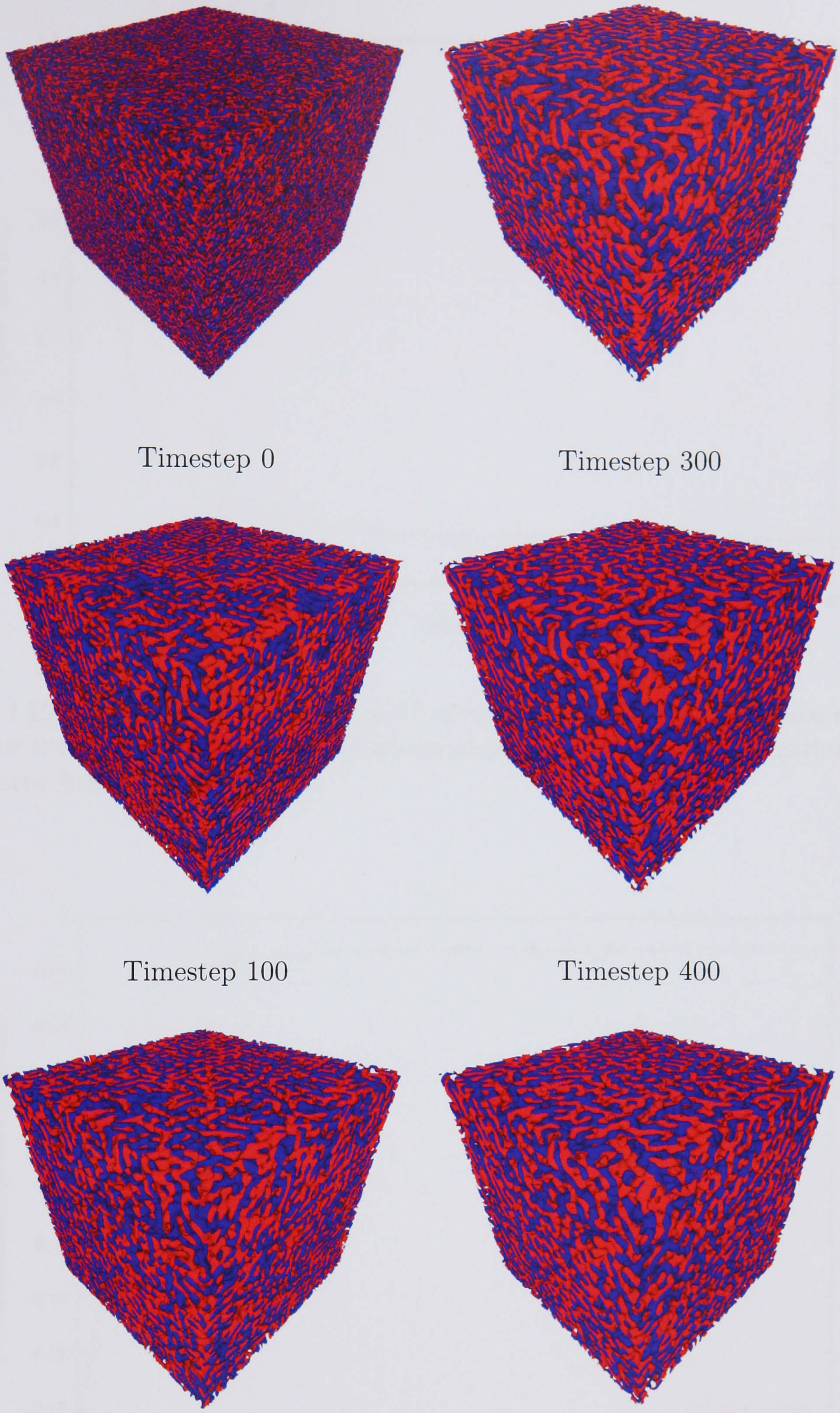


Figure 4.18: The oil-water interface ($\phi = 0$) during early-time evolution of a parameter set 8 system running on a 128^3 lattice.

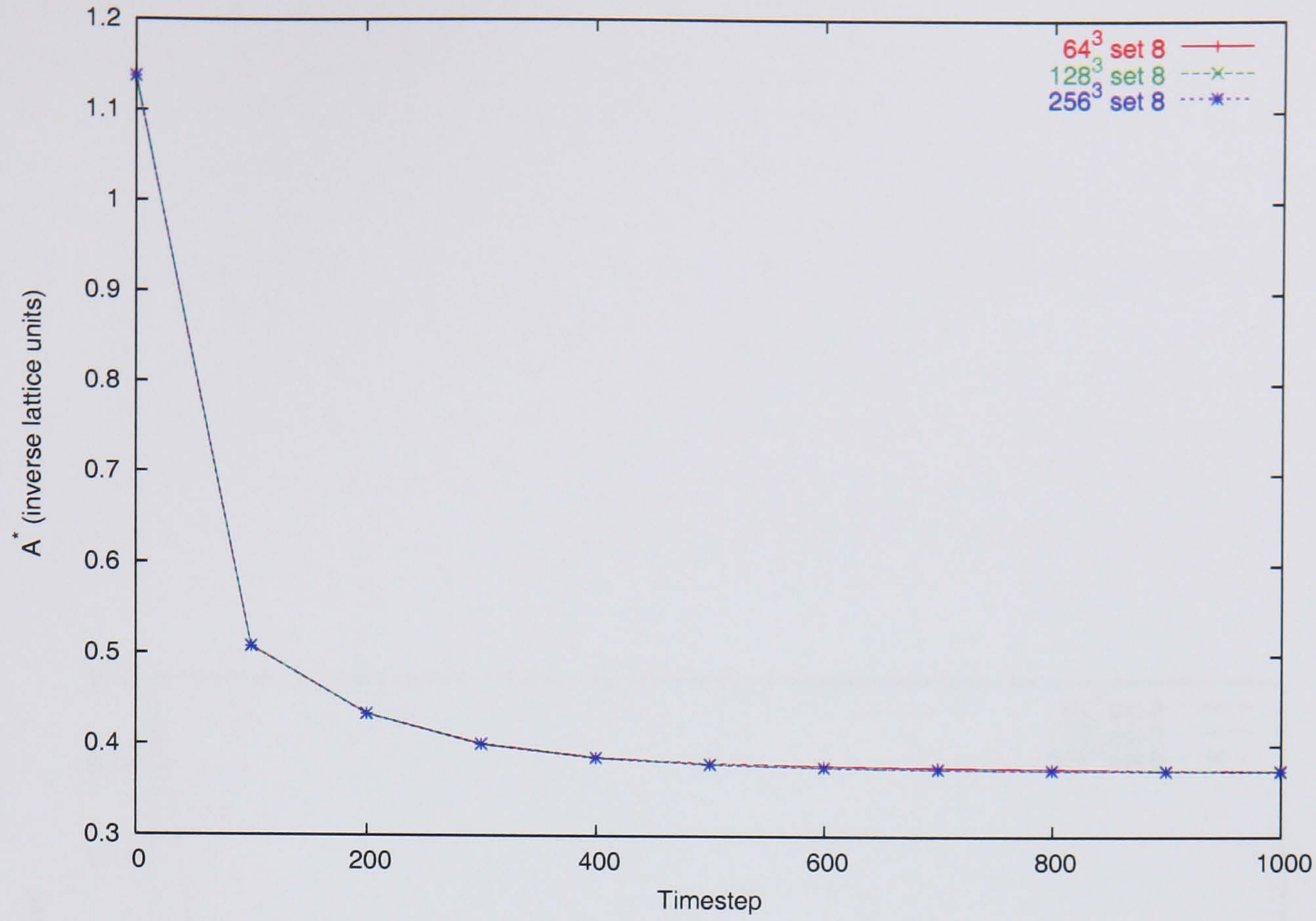


Figure 4.19: Normalized area $A^* = A/V$ of the $\phi = 0$ oil/water interface at early times for three parameter set 8 simulations, showing very close superposition of the scaled area from each simulation.

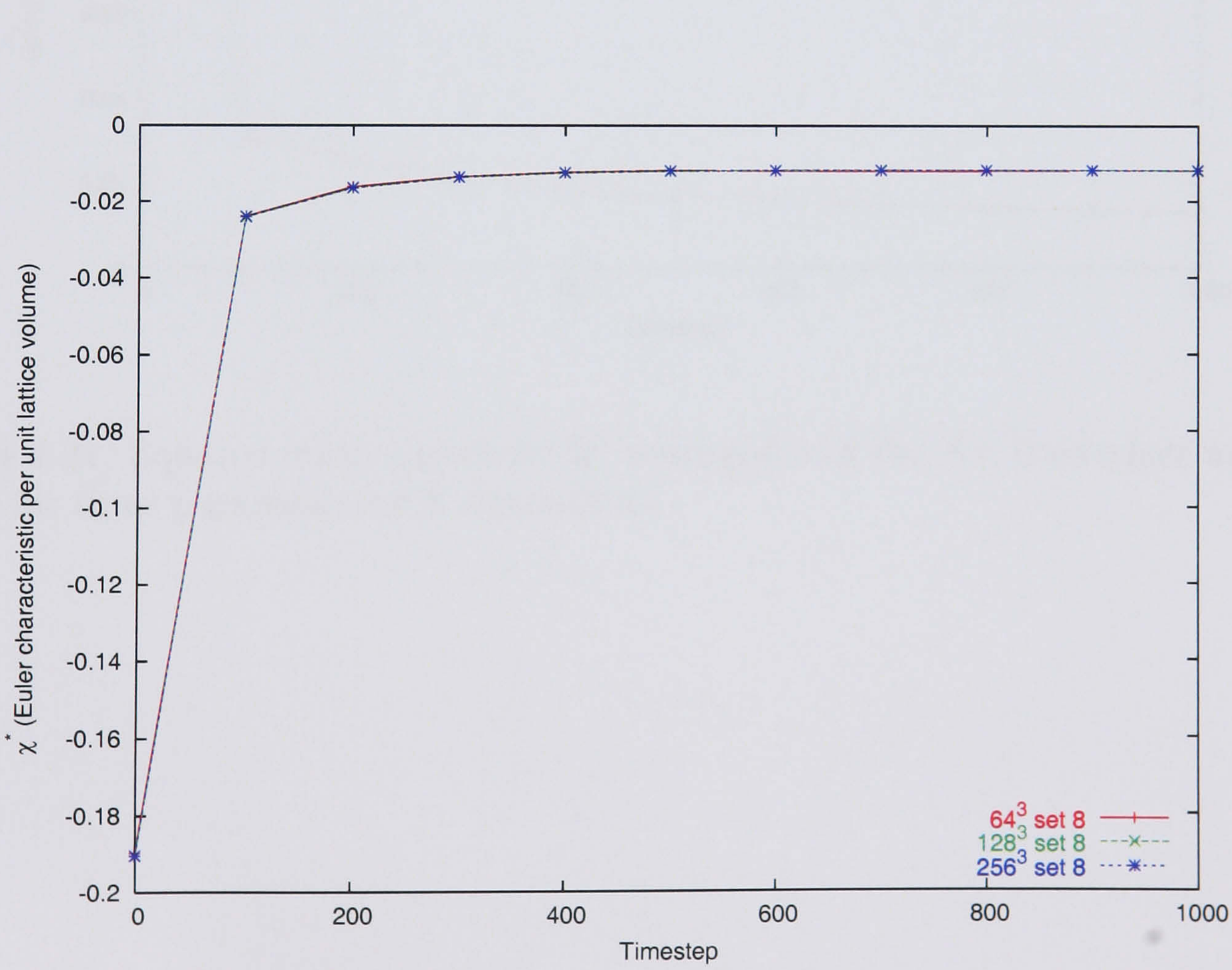


Figure 4.20: Normalized Euler characteristic $\chi^* = \chi/V$ at early times with parameter set 8.

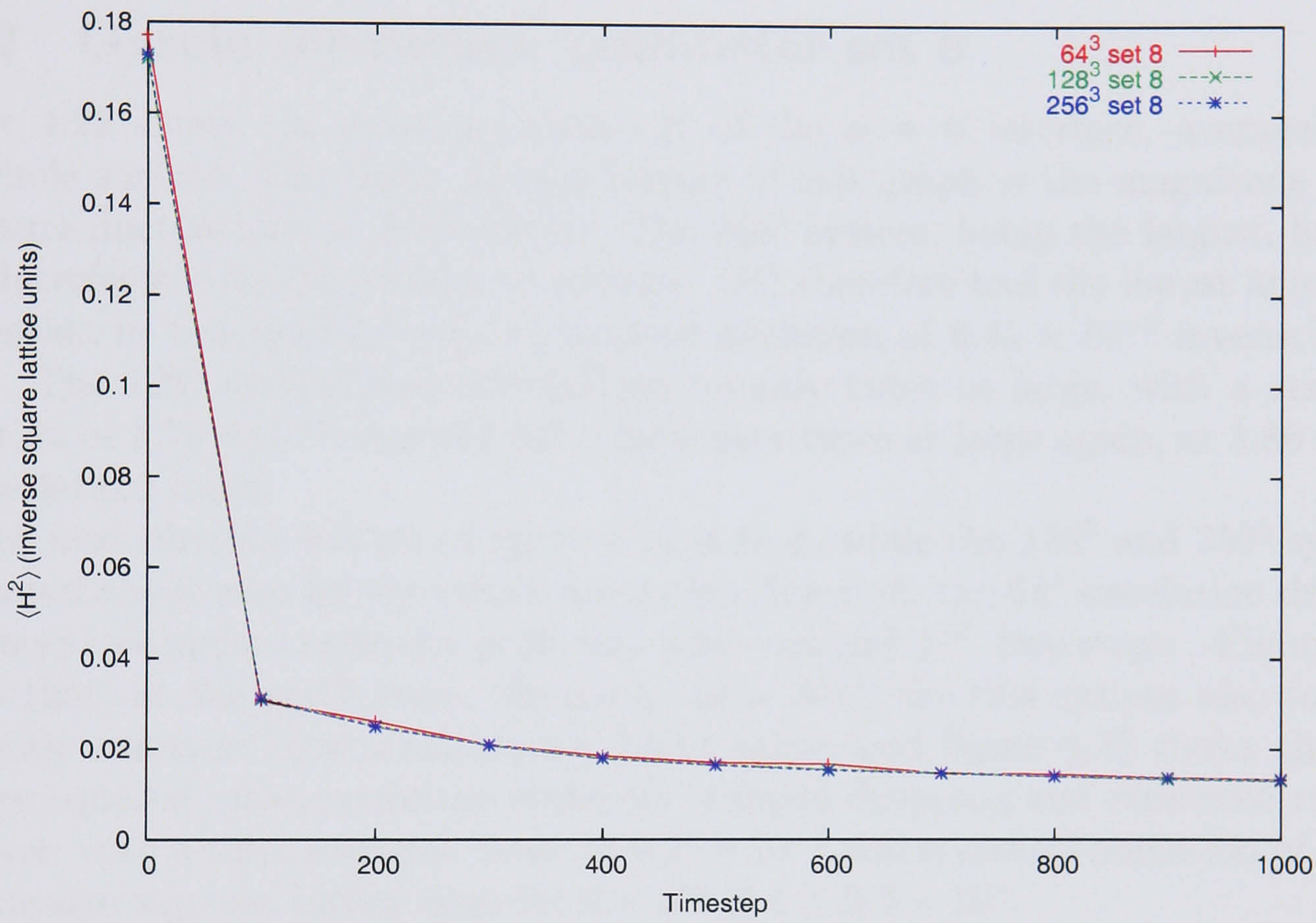


Figure 4.21: Squared mean curvature H^2 averaged over the $\phi = 0$ interface at early times for three parameter set 8 simulations.

Figure 4.19 shows a rapid drop in the area of the oil-water interface over the first thousand timesteps, as the domains formed; figure 4.21 shows that the squared mean curvature $\langle H^2 \rangle$ also rapidly dropped during this régime.

It should be noted that, despite what one might (naïvely) expect from the dimensions, the interfacial area A of the system scales with the system volume V, rather than $V^{2/3}$. This is because A is not the area *of* the volume V, but rather of a structure (the interface) which is packed inside it. Therefore, the scaled area A^* collapsed onto roughly the same graph for all system sizes.

The Euler characteristic χ of the $\phi = 0$ interface dropped in magnitude, as seen in figure 4.20, indicating that the (initially highly tortuous) interface rapidly became less interconnected. In all of these figures, the plots for the three different systems are very close, indicating that the simulated physical behaviour was independent of the lattice size over this period.

4.6.2 Gyroid formation: parameter set 9

Figure 4.22 shows the mean curvature H of the $\phi = 0$ interface, averaged over the whole surface. The most obvious feature of this graph is the magnitude of the curvature fluctuations in each system. The 256^3 system, being the largest, had the most interfacial area over which to average: $\langle H \rangle$ therefore had the lowest amount of fluctuation in this system, with a standard deviation of 8.41×10^{-5} inverse lattice units. The 128^3 system had fluctuations roughly twice as large, with a standard deviation of 1.70×10^{-4}, and the 64^3 a little over twice as large again, at 3.86×10^{-4} inverse lattice units.

The next obvious feature of figure 4.22 is that, while the 128^3 and 256^3 systems fluctuated about zero for the entire simulation duration, the 64^3 simulation dropped to a negative-mean-curvature geometry after around 10^5 timesteps. Figure 4.24 shows that, at the same time, the scaled area $A^*(t)$ for this system also took on a roughly constant (and anomalously high) value, and figure 4.23 shows that the average squared mean curvature suddenly stopped dropping and remained roughly constant, with a time-averaged mean of 9.25×10^{-4} and standard deviation of 5.45×10^{-6} inverse squared lattice sites for $9 \times 10^4 \leq t \leq 2.5 \times 10^5$.

A sudden halt in the evolution of a property such as the area is usually a sign that the system has collapsed into a stable state: figure 4.25 demonstrates that this was the case. By timestep 10^5, the system had reached the finite-size-effect régime and formed a regular structure, filling the entire simulation lattice, apart from two point defects. This liquid crystal structure had Bravais lattice vectors

$$\begin{aligned} \mathbf{a} &= (\ 64/7\ ,\ 0\ ,\ 0\ ) \\ \mathbf{b} &= (\ 128/91\ ,\ 128/13\ ,\ 128/91\ ) \\ \mathbf{c} &= (\ 0\ ,\ 0\ ,\ 64/7\ ). \end{aligned} \tag{4.28}$$

Looking along the Y axis, the structure reconnected with itself through the periodic boundary conditions, moving half a unit cell in each of the X and Z directions after traversing the lattice. There were seven unit cells in the X and Z directions, and six and a half in the Y direction: a topologically perfect gyroid with this many unit cells would have an Euler characteristic of -2548. The simulated structure had $\chi = -2540$ due to the presence of the point defects. The skewing of the "gyroid" in the X and Z directions and consequent rhombohedral (rather than cubic) liquid crystal structure meant that the oil-water interface did not form a zero-mean-curvature

gyroid structure, but a skewed gyroid with, on average, negative mean curvature. It should be noted that the production of a gyroid with this orientation is a reassuring sign that the production of a gyroid mesophase is not a simulation artifact due to some anisotropy induced by the lattice, in which case such misalignment would not be expected.

The normalized area, plotted in figure 4.24, changed relatively little compared to the rapid initial drop in figure 4.19. The 64^3 system sat at an abnormally high value of normalized area, again due to its skewed gyroid morphology. Figure 4.26 shows the Euler characteristic per lattice site, which in all systems dropped to a value of around $\chi^* = -0.00895$ per site. For a perfectly gyroidal system, this would imply a gyroid lattice parameter of $(-8/\chi)^{1/3} \simeq 9.6$ lattice sites, agreeing well with the lattice parameter lying between 9 and 10 observed in the simulations (equations 4.28).

A closer examination of figure 4.22 shows that the average mean interfacial curvature in the 128^3 system also fluctuated around a slightly negative value, rather than zero: for $3 \times 10^5 \leq t \leq 10^6$, $\langle H \rangle$ has a mean of -8×10^{-5} and a standard deviation of 9×10^{-5} inverse lattice lengths. Figure 4.23 shows that this simulation appeared to succumb to finite-size effects around timestep 3×10^5, undergoing a curious decrease in interfacial area (figure 4.24), and connectivity (figure 4.26) for $2.3 \times 10^5 \lesssim t \lesssim 4 \times 10^5$.

Figure 4.29 shows the $\phi = 0$ isosurface, coloured red on the oil-majority side and blue on the water-majority side, at timestep 2.5×10^5. The region shown lies at the interface between the two domains; the domain wall is shown running vertically through the centre of the diagram. Careful examination of the channels running through the diagram shows that, while the two domains contained gyroids with roughly the same orientation, the domains had opposite chirality: red channels spiral clockwise away from the viewer in the right hand domain, but anticlockwise away from the viewer in the left hand domain. Since the model used is oil-water symmetric, there should be no preference for, say, oil channels to be left-handed, so the existence of chiral domains is to be expected.

Figure 4.28 shows volume renderings highlighting regions of this simulation which had both low values of the order parameter and low values of the order parameter gradient, a technique which was observed to show up regions containing defects or walls between gyroid domains. At timestep 10000, the gyroid structure had not emerged, but by timestep 40000, multiple gyroid domains were visible. These domains were observed to merge, until only two domains were left in the system, around timestep 200000. One of the domains then collapsed over the next hundred thousand timesteps, leaving behind two columnar defect regions which persisted for as long as the simulation continued. The period of time over which this domain collapsed is the same as the duration of the blip in connectivity and interfacial area. The decrease in interfacial area may have been due to the formation of large blobby structures in the domain wall. Figure 4.27 illustrates this: the bottom-right corner of the image shows a domain with red channels spiralling in a clockwise direction away from the viewer; the top-left region corner contains a domain with the red channels spiralling anticlockwise. At the domain wall in the centre lie several amorphous objects which aggregate a relatively large amount of oil or water into a single mass, rather than spreading it out into slender almost-cylindrical channels as happens in the gyroid regions. Such structures have a smaller surface area and connectivity than the gyroid regions, giving rise to the anomaly in figures 4.24 and 4.26.

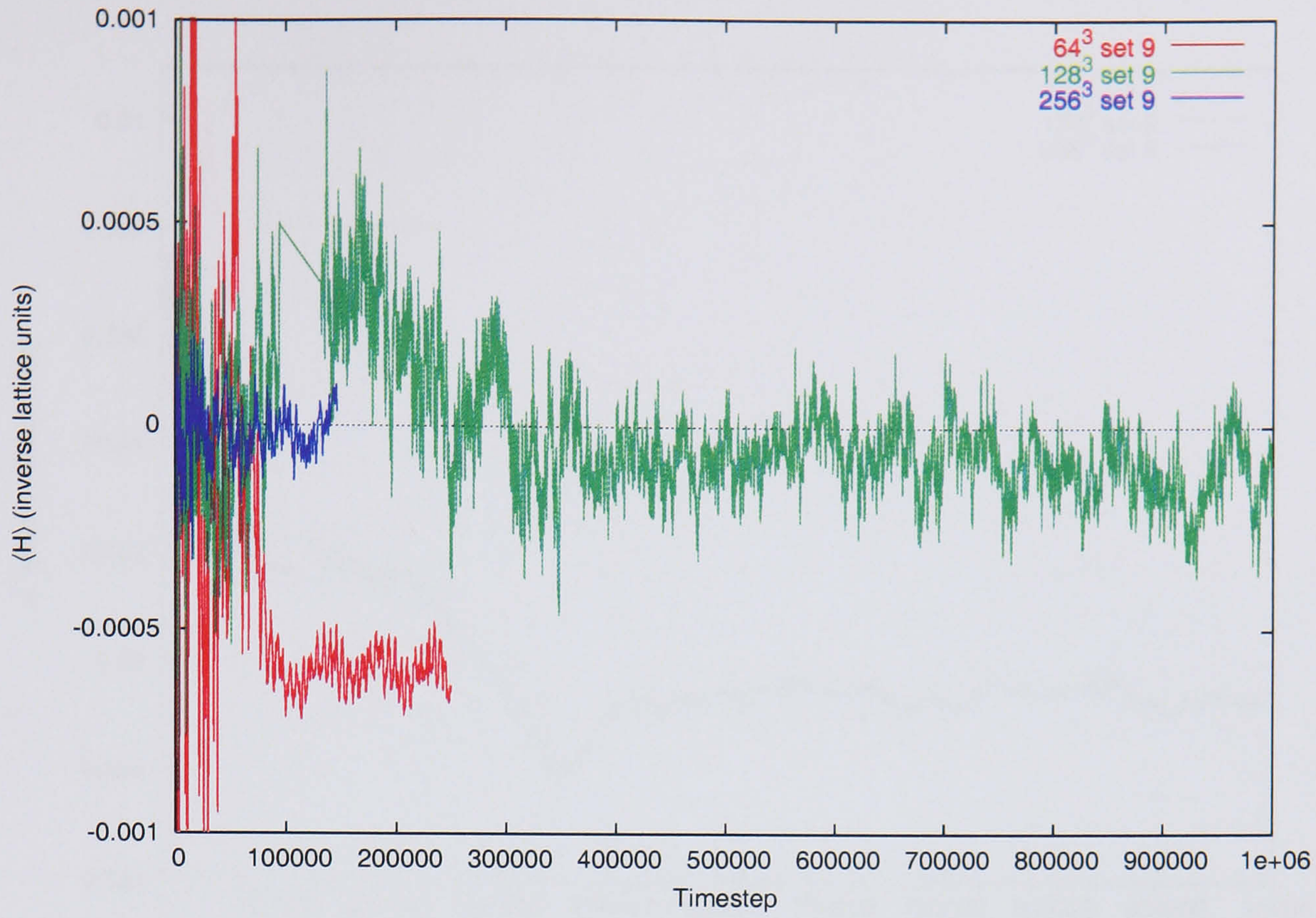


Figure 4.22: Mean curvature H averaged over the $\phi = 0$ surface for parameter set 9, for three different simulation lattice sizes.

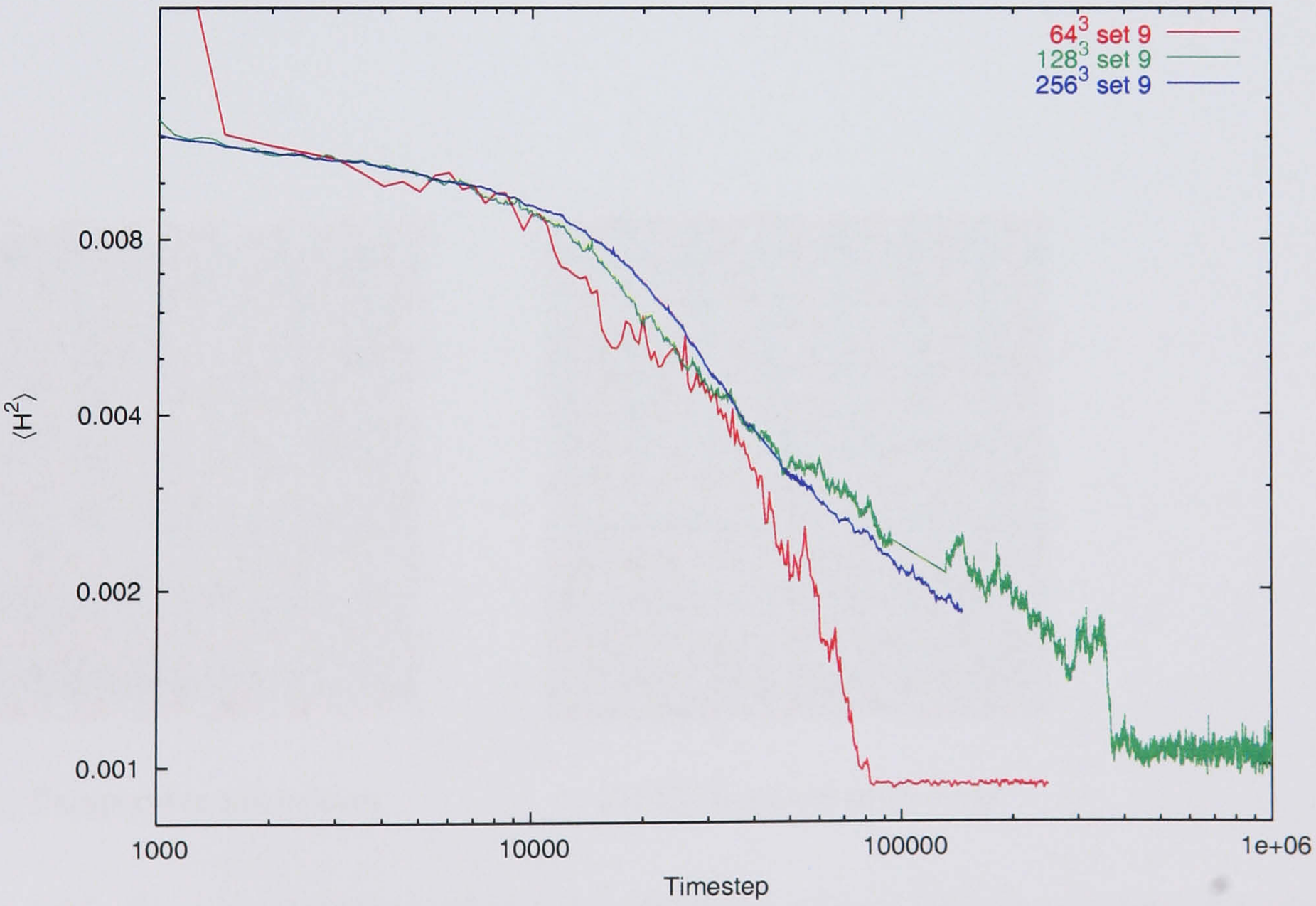


Figure 4.23: Squared mean curvature H^2 averaged over the $\phi = 0$ surface for parameter set 9, for three different simulation lattice sizes.

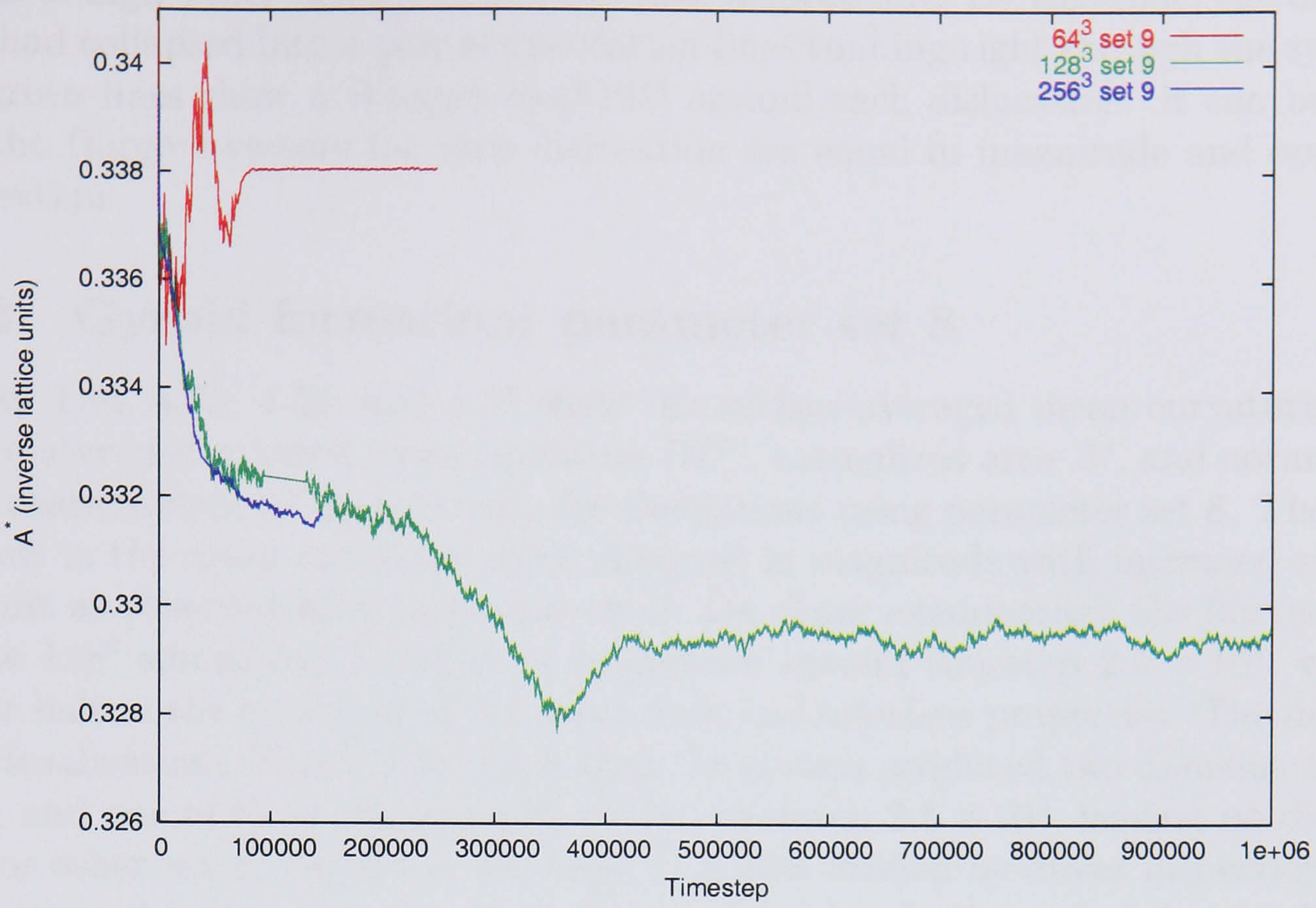
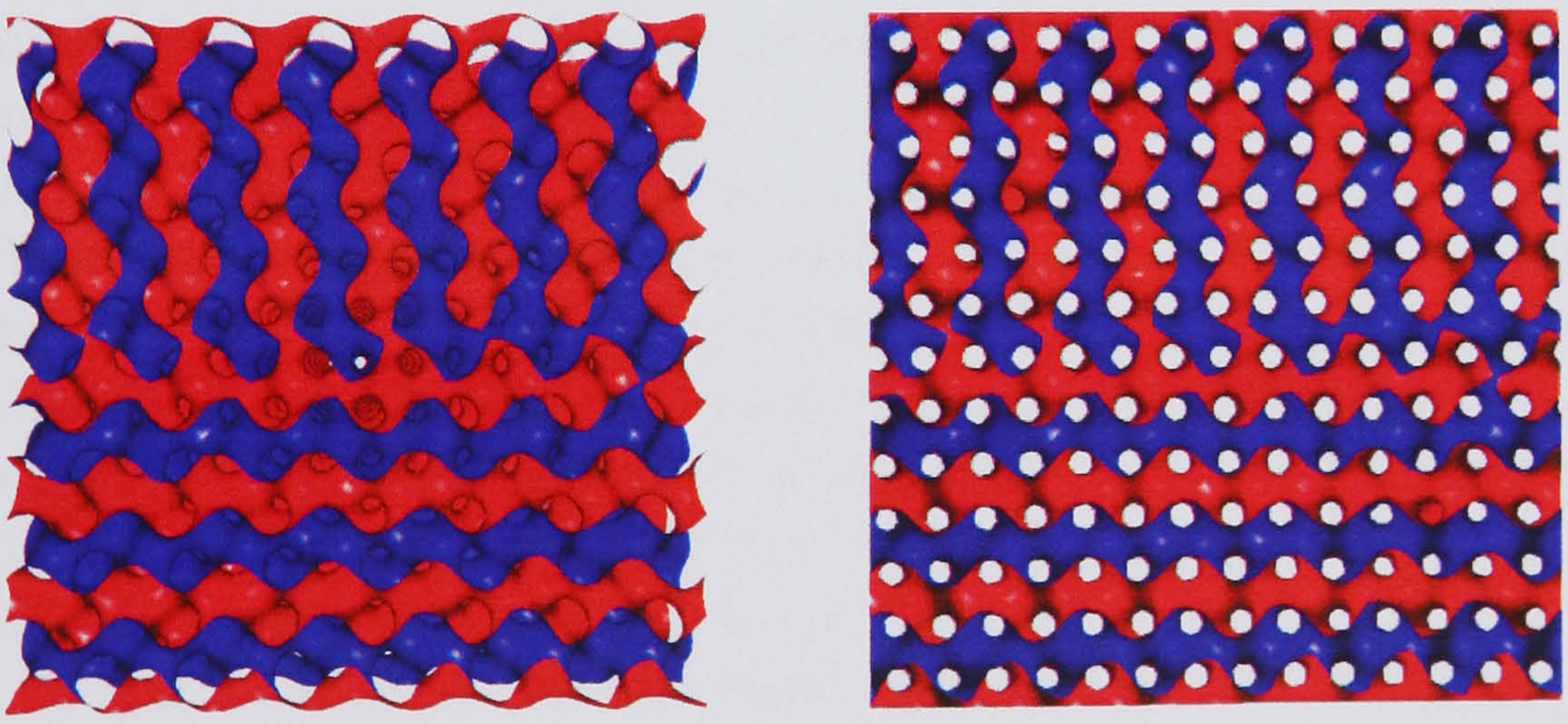


Figure 4.24: Normalized area $A^* = A/V$ of the $\phi = 0$ surface for parameter set 9 for three different simulation lattice sizes. The 64^3 simulation rapidly reaches a finite size induced perfect gyroid state.



(a) Perspective projection (b) Orthogonal projection

Figure 4.25: Two projections of the $\phi = 0$ surface for a 64^3 simulation of parameter set 9. While there appears to be a dislocation running horizontally, this is a rendering artifact caused by the liquid crystal axes running at an angle to the simulation box.

Figure 4.30 shows a volume rendering of the order parameter field at timestep 5×10^5, with the transfer function chosen to highlight regions of high oil density and regions of high water density in red and blue respectively. By this time, the domain walls had collapsed into a pair of dislocation lines running right through the system. The green lines show a Burgers loop[195] around each dislocation: it can be seen that the Burgers vectors for each dislocation are equal in magnitude and opposite in direction.

4.6.3 Gyroid formation: parameter set 8

Figures 4.34, 4.35, 4.36, and 4.37 show the surface-averaged mean curvature $\langle H \rangle$, surface-averaged squared mean curvature $\langle H^2 \rangle$, normalized area A^*, and normalized Euler characteristic χ^* respectively, for simulations using parameter set 8. The fluctuations in the mean curvature again dropped in magnitude with increased system size, just as observed with parameter set 9. On closer examination, the fluctuations for the 128^3 simulation dropped in magnitude around timestep 2.5×10^5, with a similar halt in the evolution of the other bulk and interface properties. The domain wall visualizations (figure 4.38) show that the system produced two domains at late times, and one of these disappeared around timestep 2.5×10^5, leaving no dislocations or other major defects in the system. It was verified by direct inspection that the last two domains were again of opposite chirality. At the end of the simulation, timestep 378500, the system was found to contain a gyroid with 15 unit cells in each direction, with an Euler characteristic of -26828, corresponding to 3353.5 gyroid unit cells. A 15^3 periodic gyroid would be expected to contain 3375 unit cells: the discrepancy is attributed to persistent point defects, which appeared as the domains formed, up to timestep 10^5, and remained unchanged for the rest of the simulation.

The 256^3 simulation, limited only by available computer time, ran until timestep 57500, at which point it was still composed of multiple chiral gyroid domains. The domains were observed to interpenetrate in a manner not unlike the morphology of single-component fluid domains during spinodal decomposition.

Figure 4.39 shows two transmission electron micrographs (TEM) of a triblock copolymer gyroid, taken from the work of Laurer *et al* [217], along with orthographic-projection volume renderings of two parameter set 8 simulations. These images all look down the (111) direction, producing a distinctive "wagon wheel" pattern. Figure 4.39a shows some irregularity in the gyroid structure, similar to that seen in the simulated 128^3 system in 4.39d, and rather exaggerated in the 256^3 system in 4.39c. While it can be seen clearly in the simulated results that the blurring of the pattern is due to the presence of differently oriented or chiral domains underneath the top layers, unfortunately the resolution of the TEM images is not quite high enough to distinguish this in the experimental results.

Figure 4.31 shows the averaged squared mean curvature plotted against time for 128^3 and 256^3 simulations of parameter sets 8 and 9. Both of the parameter set 8 systems appeared to take on fairly close power-law behaviour (seen as straight lines on the log-log graphs) after timestep 10^4, when the system recognizably contained gyroid regions. The parameter set 9 systems also did this, but took longer to do so: the value of $\langle H^2 \rangle$ for the set 9 systems was somewhat larger than that of the set 8 systems, until around timestep 5×10^4, at which point they appeared to show the same behaviour. The 128^3 systems eventually deviated from the power-law behaviour as finite-size effects set in, around timestep 2×10^5 for set 8, and 3×10^5

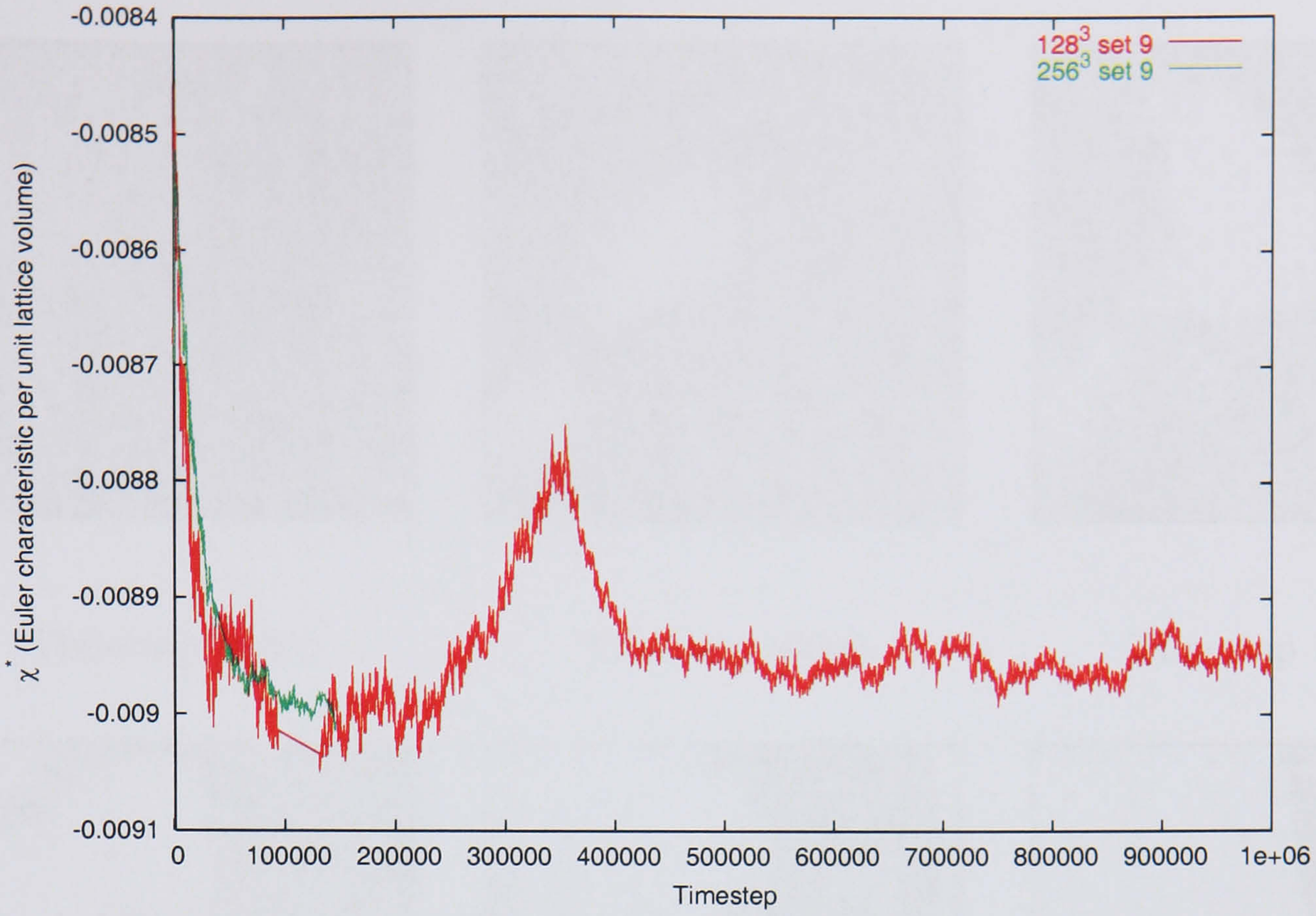


Figure 4.26: Normalized Euler characteristic $\chi^* = \chi/V$ of the $\phi = 0$ surface for parameter set 9.

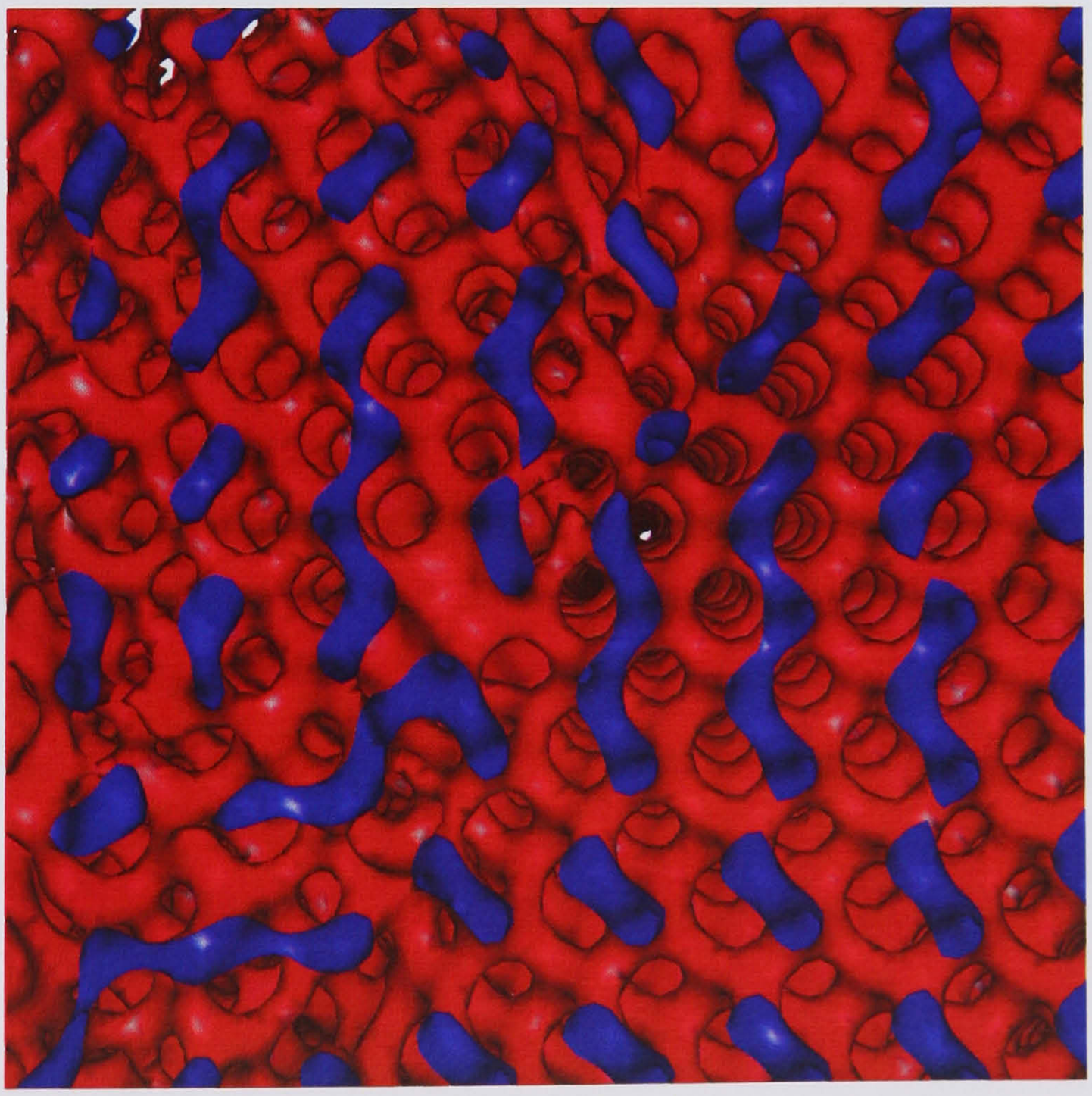


Figure 4.27: The $\phi = 0.3$ isosurface of the 128^3 simulation of parameter set 9 at timestep 3.5×10^5, showing blob structures in a domain wall.

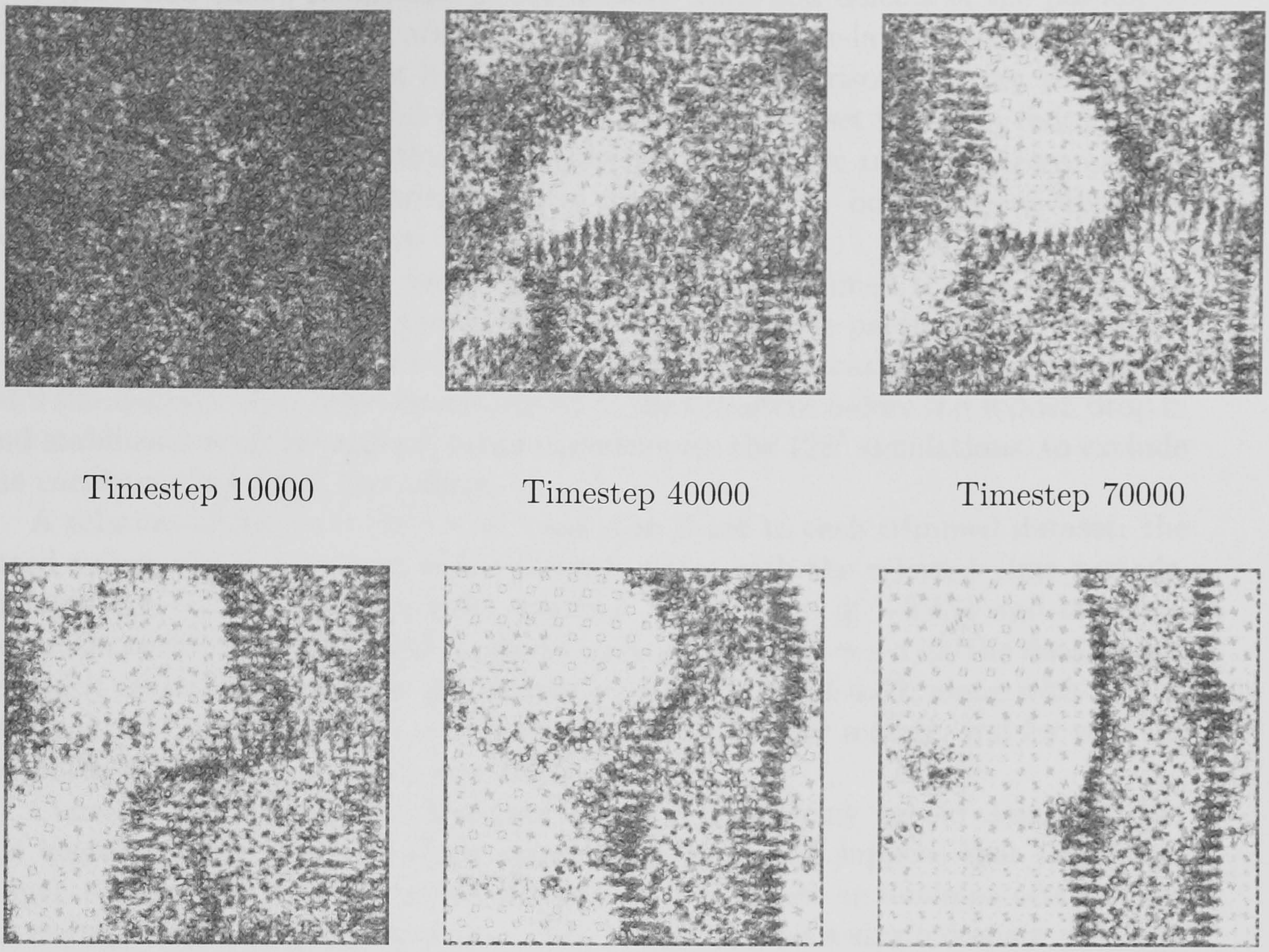


Figure 4.28: Orthographic-projection volume rendering highlighting domain walls for a 128^3 simulation using parameter set 9, showing the late-time collapse of the last domain into a pair of line dislocations. Periodic boundaries are enforced; one of the dislocations straddles an edge of the simulation box.

for set 9.

Figure 4.33 shows renderings of the domain walls and defects of the parameter set 9 system at points before and just after it reached power-law behaviour, with the defects for the parameter set 8 system plotted for comparison. During the period when the set 8 system showed power-law scaling but the set 9 system did not, the latter appeared to have a rather higher density of defective regions, shown as dark regions in the volume renderings. After timestep 50000, both systems appeared clearly separated into domains.

The datasets for $\langle H^2 \rangle(t)$ for each simulation were trimmed to only include the timesteps after 10000, when gyroid structures form, for the parameter set 8 simulations, and the timesteps after 50000, after clear domain formation, for the parameter set 9 simulations; they were also restricted to the timesteps before the sudden drop in and stabilization of the squared mean curvature for the 128^3 simulations, to exclude the corresponding finite size effects.

A relation of the form $\langle H^2 \rangle = at^n$ was then fitted to each trimmed dataset: the fitted values of the exponent n are printed, along with the relevant time periods, in table 4.3. The fitted exponents together had a mean of -0.481 and standard deviation of 0.037. This is quite close to an exponent of $n = -1/2$; the data points to which these fits were made are plotted together in figure 4.31, along with a $t^{-1/2}$ power law for comparison. A possible reason for power-law scaling, and for why the exponent should be $-1/2$, is made below.

Consider a large system of volume V, containing many gyroid domains. Let the length scale of a typical single domain be $L_1(t)$, and suppose that the gyroid length scale grows as a power-law of time, $L_1 \sim t^n$, just as domains often do in, for example, spinodal decomposition. The surface area of a single domain will then scale as $A_1 \sim t^{2n}$, and the volume of a single domain as $V_1 \sim t^{3n}$.

The total number N of domains in the system is V/V_1, so $N \sim t^{-3n}$. The total surface area A of domains is proportional to both the number of domains and the typical surface area of a single domain, so therefore $A \sim t^{-n}$. If the domain walls all have roughly the same width λ, then the total volume of the domain walls scales as $\lambda A \sim t^{-n}$. Therefore, the contribution of the domain walls to any extensive property of the system would also be expected to scale as t^{-n}.

A gyroid is a minimal surface, so one would expect the contribution to the average squared curvature of the system from the well-formed gyroid regions inside domains to be small. On the other hand, the domain walls are non-gyroidal, and have nonzero mean curvature: if there is a roughly constant contribution to $\langle H^2 \rangle$ per unit volume of domain wall, then $\langle H^2 \rangle$ could also be expected to scale as t^{-n}. If this is the case, then that would further suggest that the length scale of the gyroid domains indeed scales as $L \sim t^{1/2}$, indicative of random-walk or diffusive behaviour.

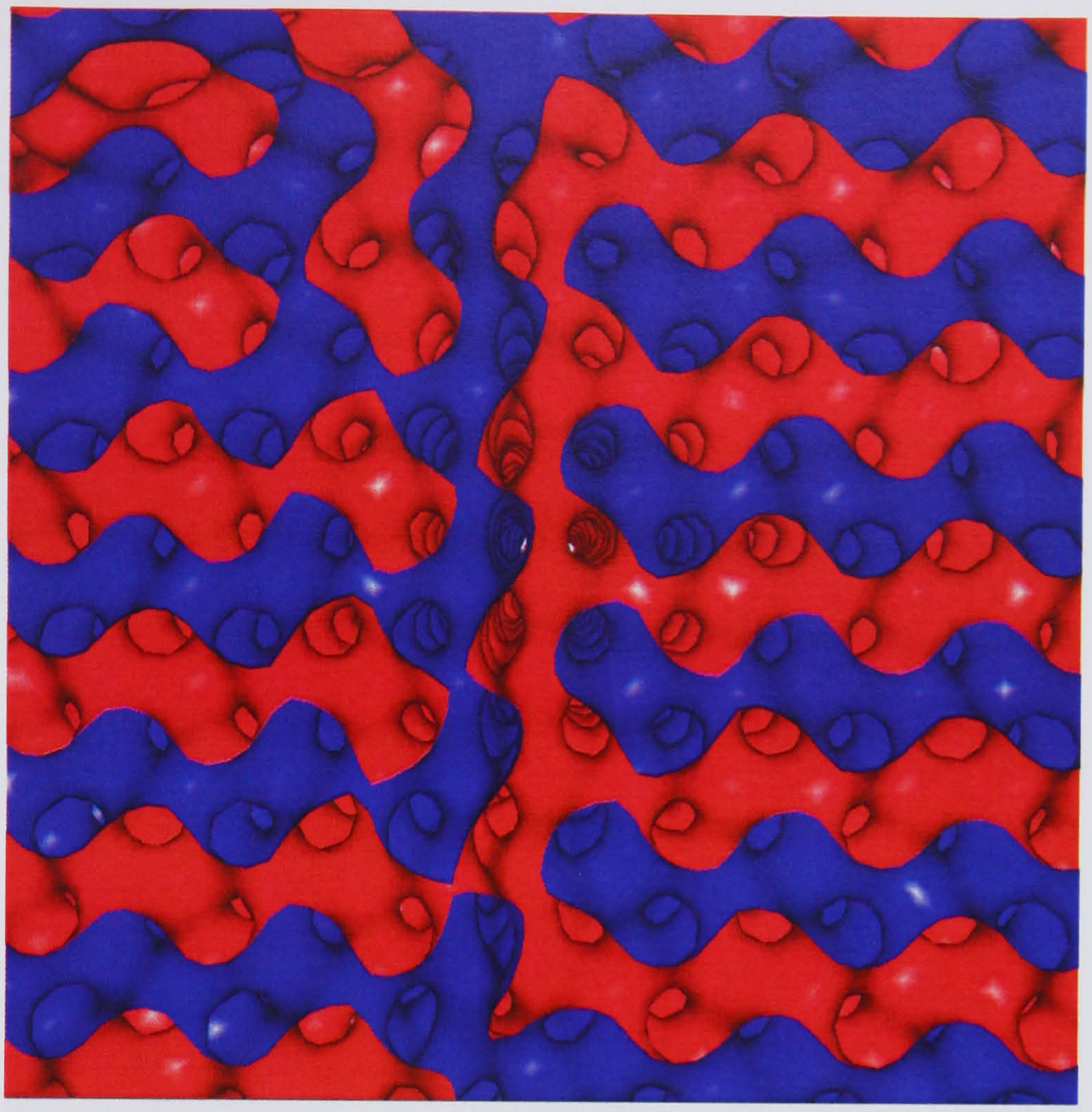


Figure 4.29: The oil-water interface at timestep 2.5×10^5 for the 128^3 simulation of parameter set 9. Note that the domains have opposite chirality.

System	$t_{\min}$	$t_{\max}$	Exponent ($\pm$ uncertainty in fit)
128^3 set 8	10000	200000	-0.474 ± 0.001
128^3 set 9	50000	300000	-0.423 ± 0.002
256^3 set 8	10000	057500	-0.5078 ± 0.0008
256^3 set 9	50000	146500	-0.519 ± 0.002

Table 4.3: Exponents n for the relation $\langle H^2 \rangle = at^n$ fitted to the squared mean curvature of four simulations between the times $t_{\min}$ when clear domains have formed and $t_{\max}$ when the finite size of either the system or the available computer resources limits the simulation.

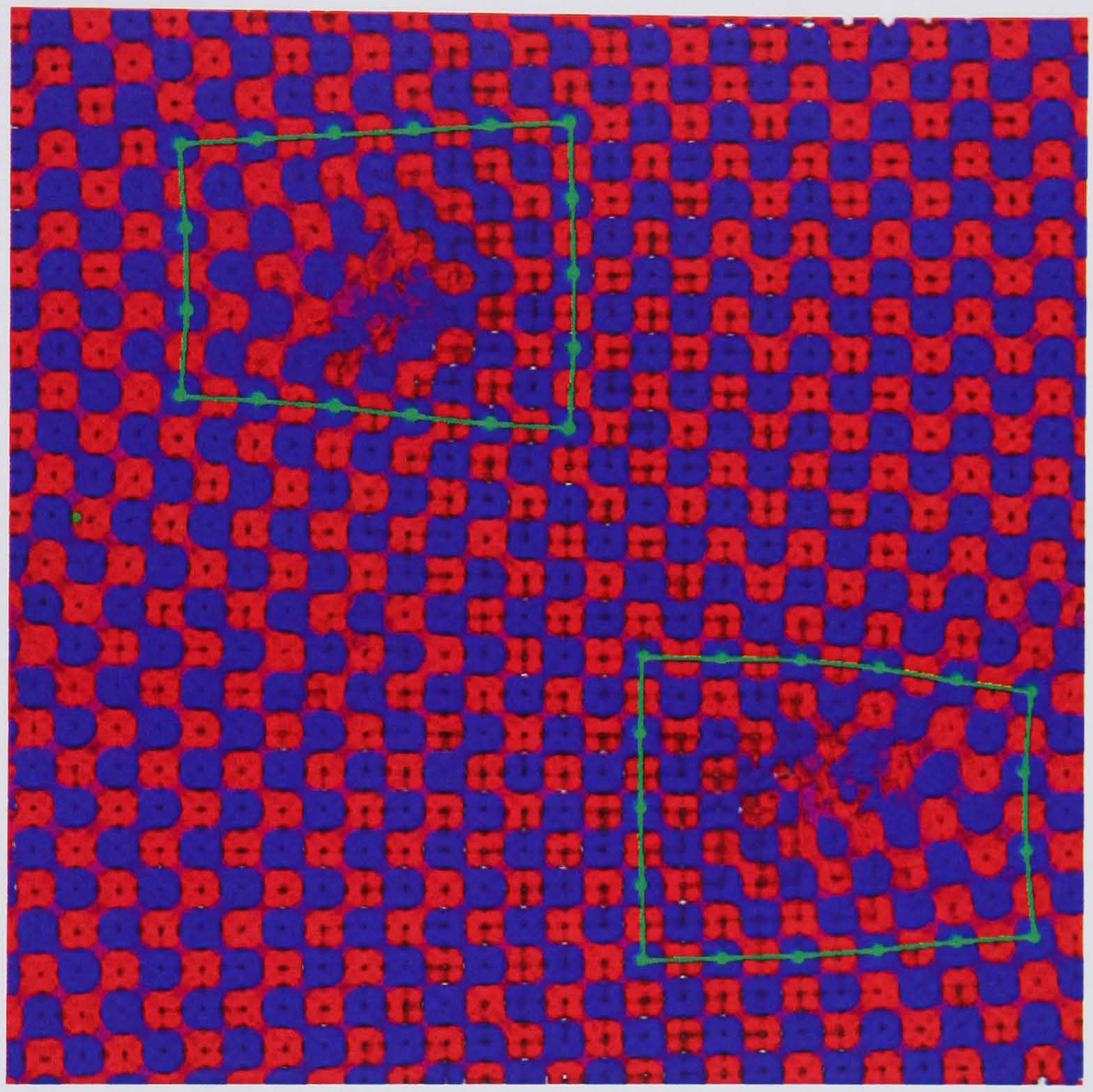


Figure 4.30: Orthographic projection volume rendering of the 128^3 simulation of parameter set 9 at timestep 5×10^5. Oil-bearing regions are highlighted in red, water-bearing regions in blue, and the interfacial regions are suppressed. The green lines show Burgers circuits around the two dislocations.

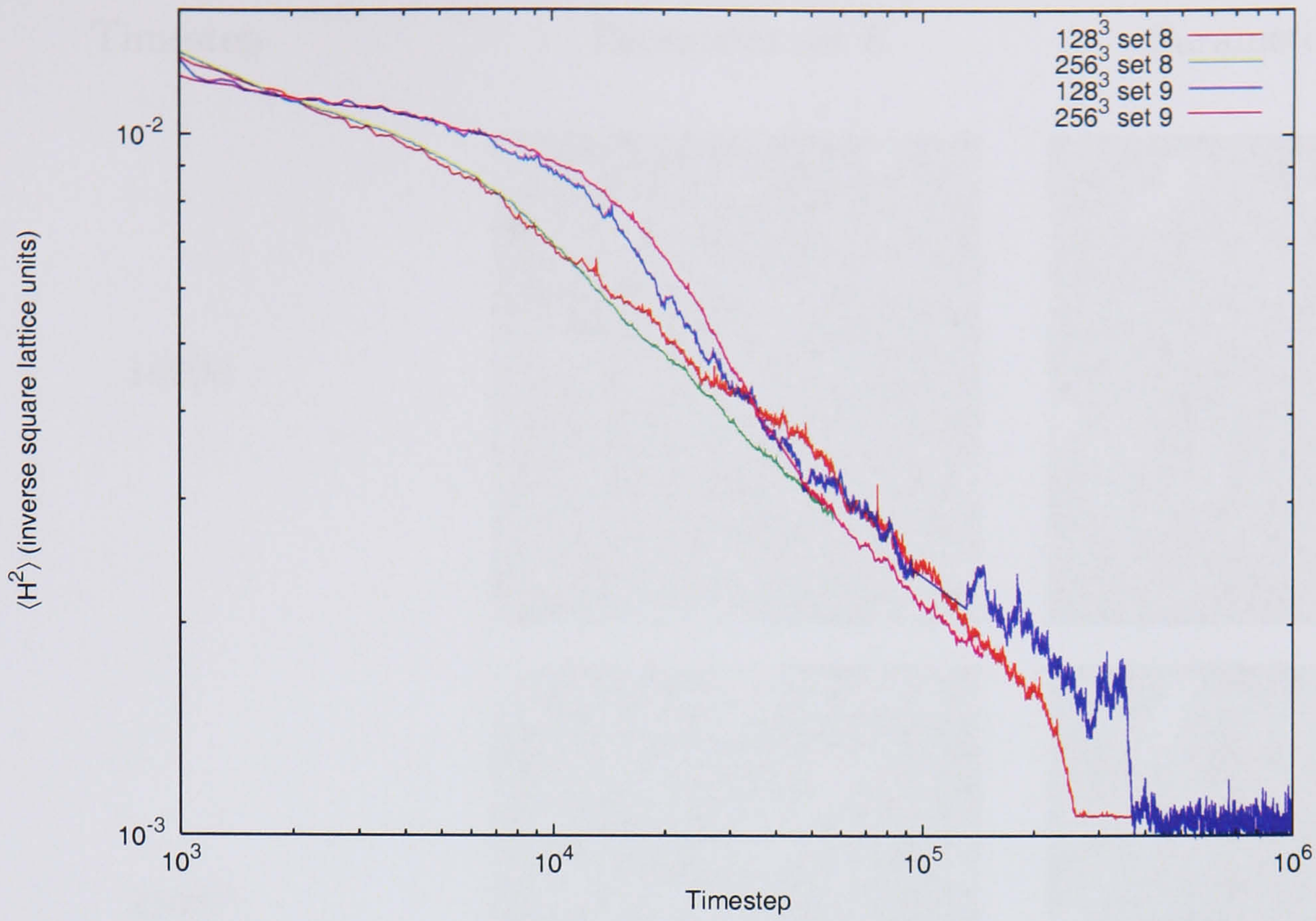


Figure 4.31: Averaged squared curvature against time for four simulations of different sizes and parameter sets. Late-time finite size effect behaviour is clear in the 128^3 simulations, where the curvature drops close to zero as a perfect minimal surface is formed.

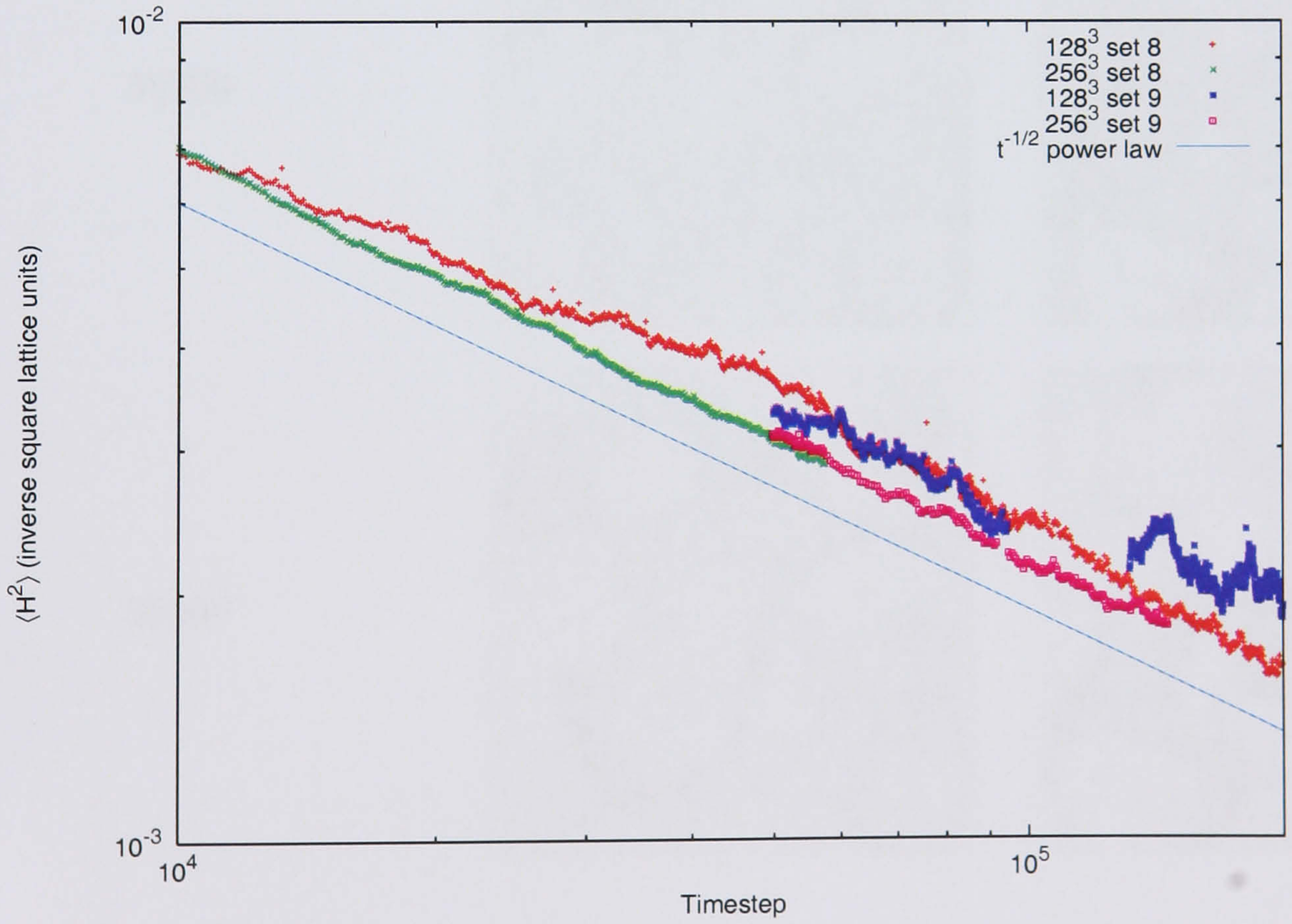


Figure 4.32: Averaged squared curvature in the power-law scaling régime for two parameter sets and sizes, with a $t^{-1/2}$ power law plotted as a guide to the eye.

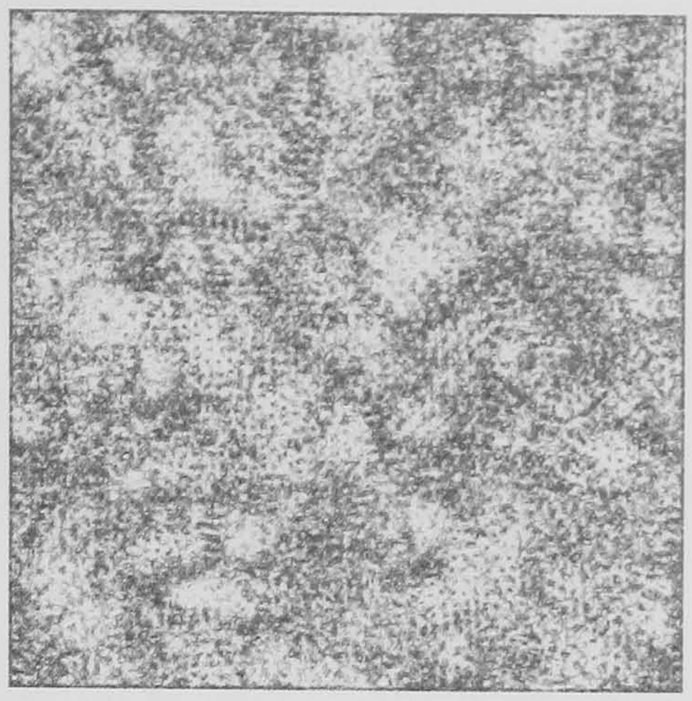
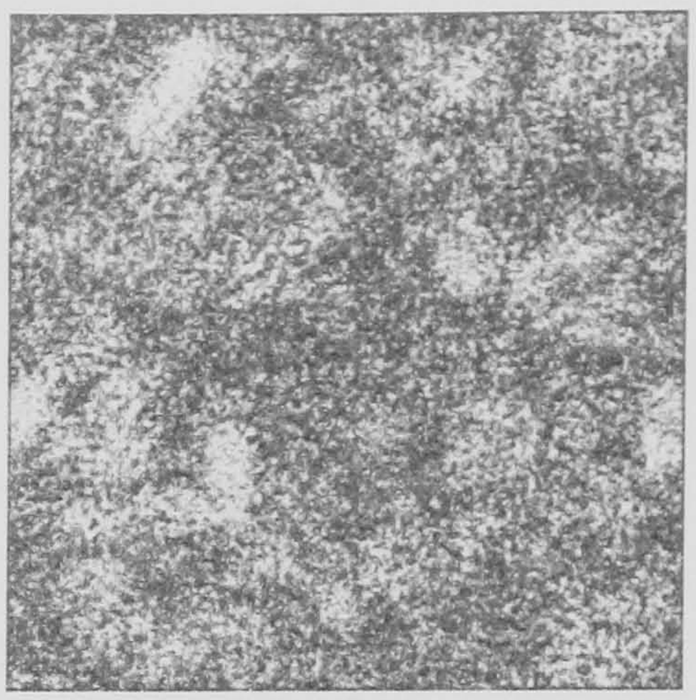
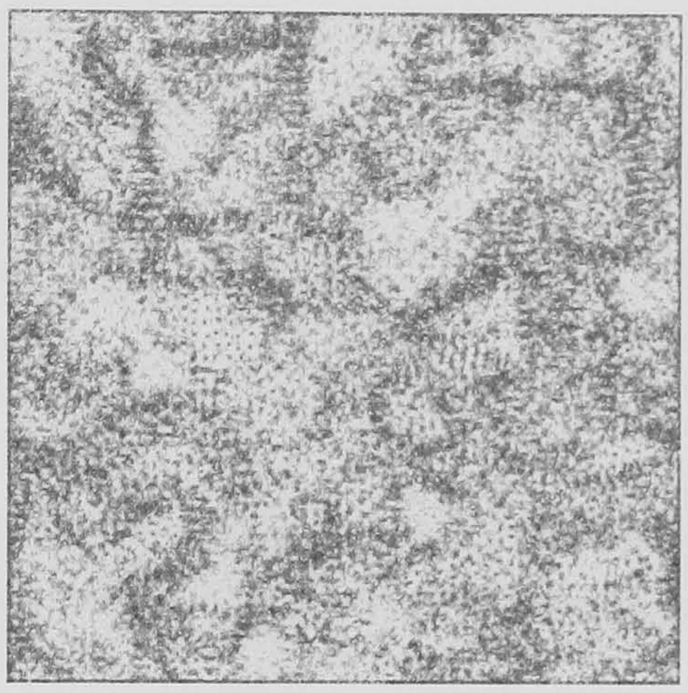
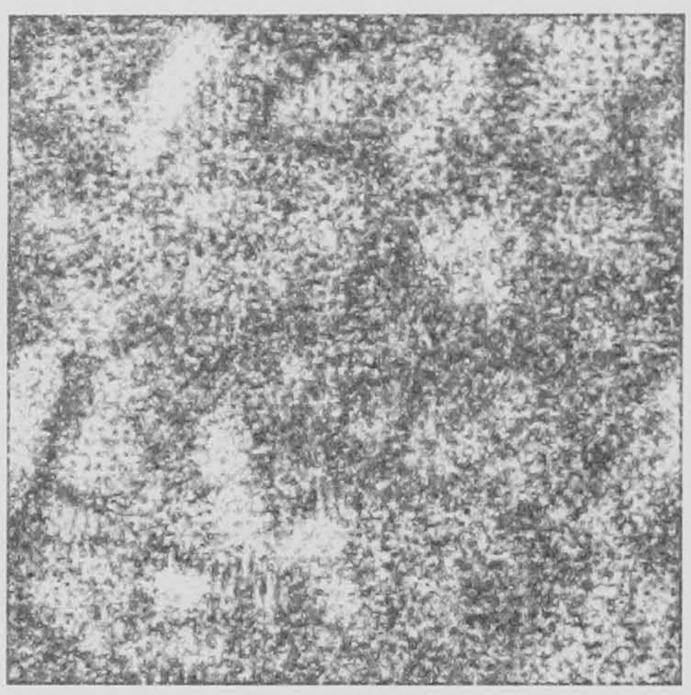
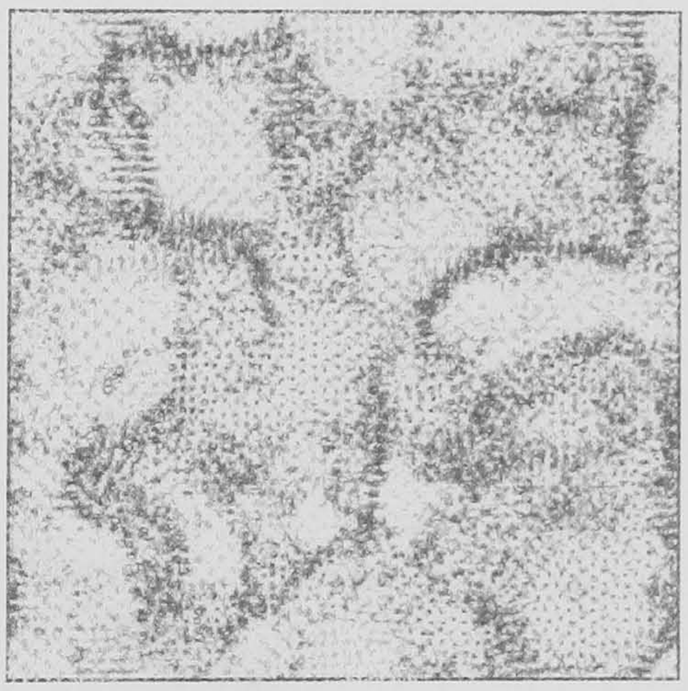
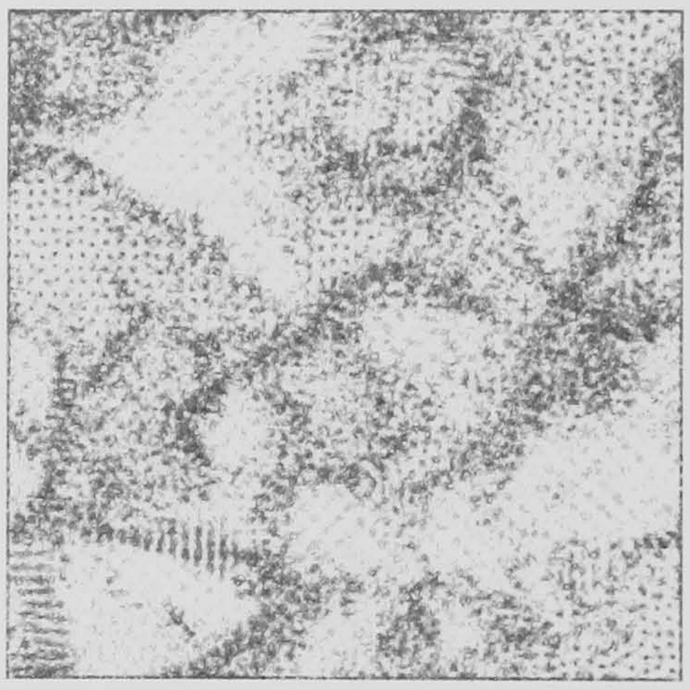
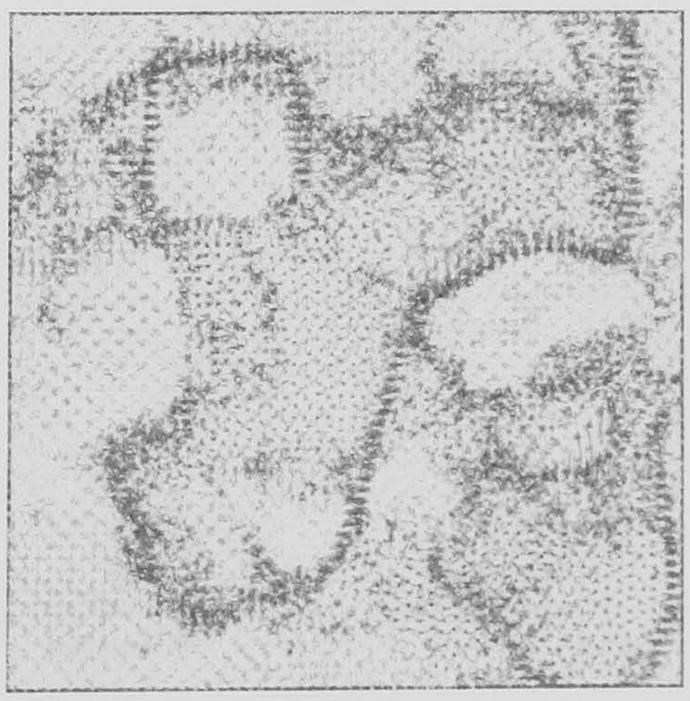
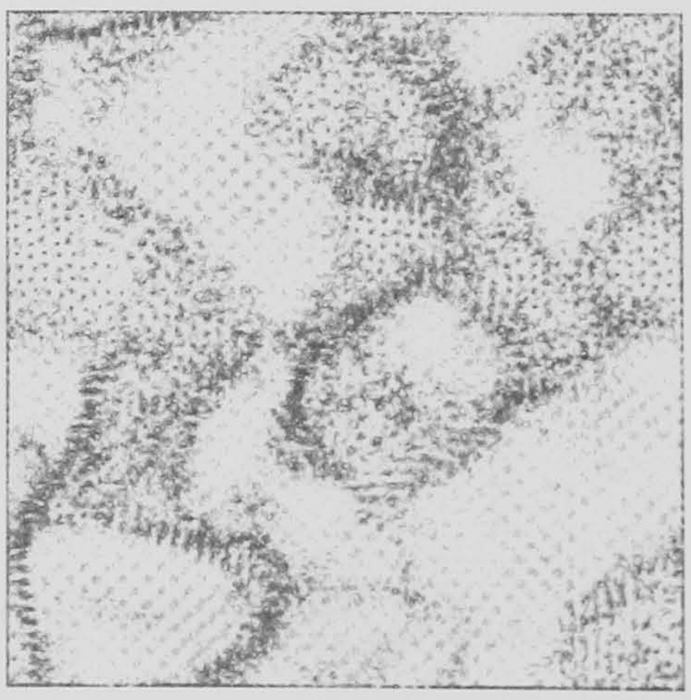
Timestep	Parameter set 8	Parameter set 9
16000		
21000		
38000		
55000		

Figure 4.33: Volume rendering of domain walls in two 256^3 systems. At earlier times, the parameter set 9 system has a larger density of defective, non-gyroidal regions. By timestep 55000, both systems are much more clearly separated into domains.

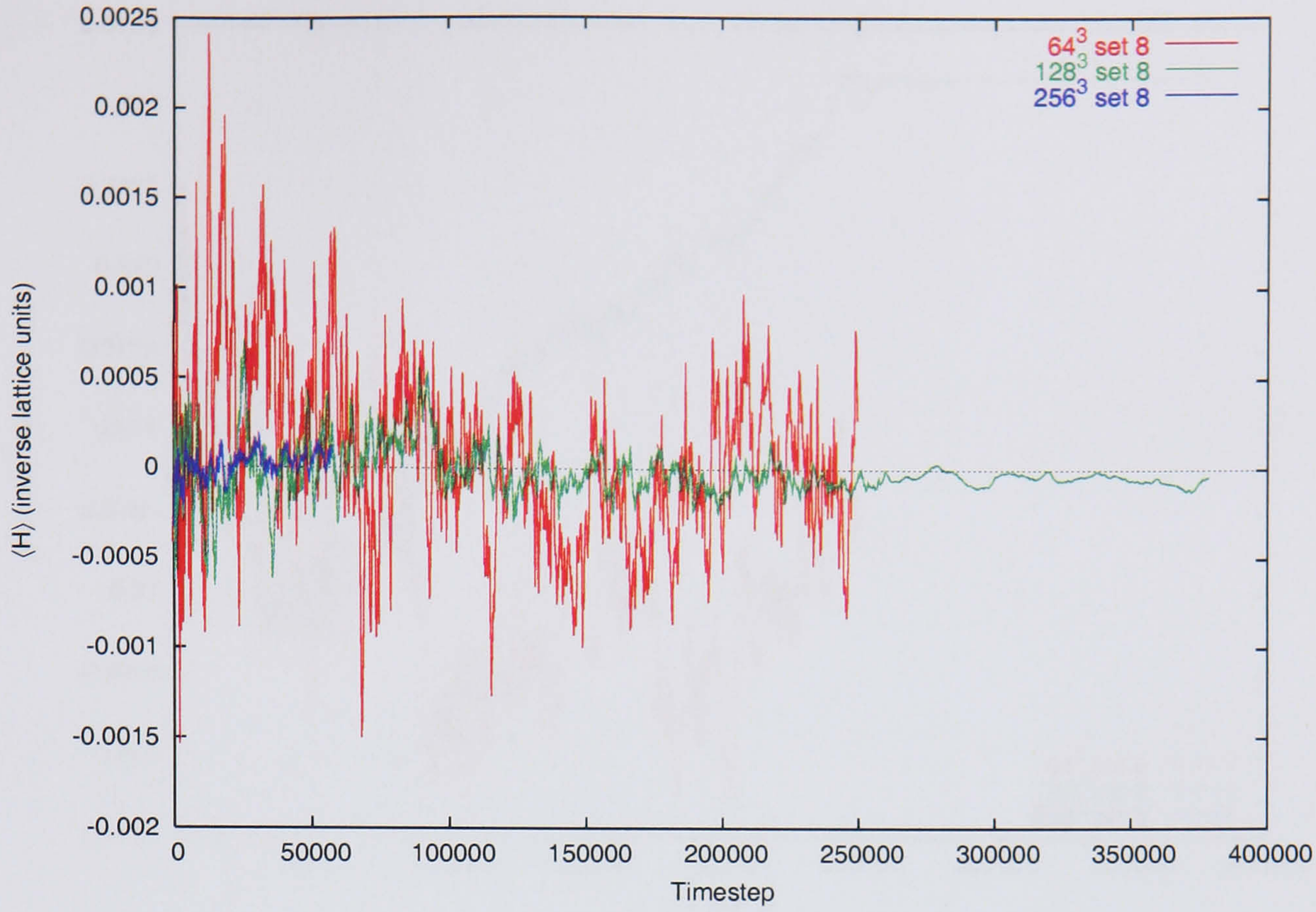


Figure 4.34: Mean curvature H averaged over the $\phi = 0$ surface for parameter set 8, for three different simulation sizes, showing that the fluctuation in mean curvature is reduced with system size.

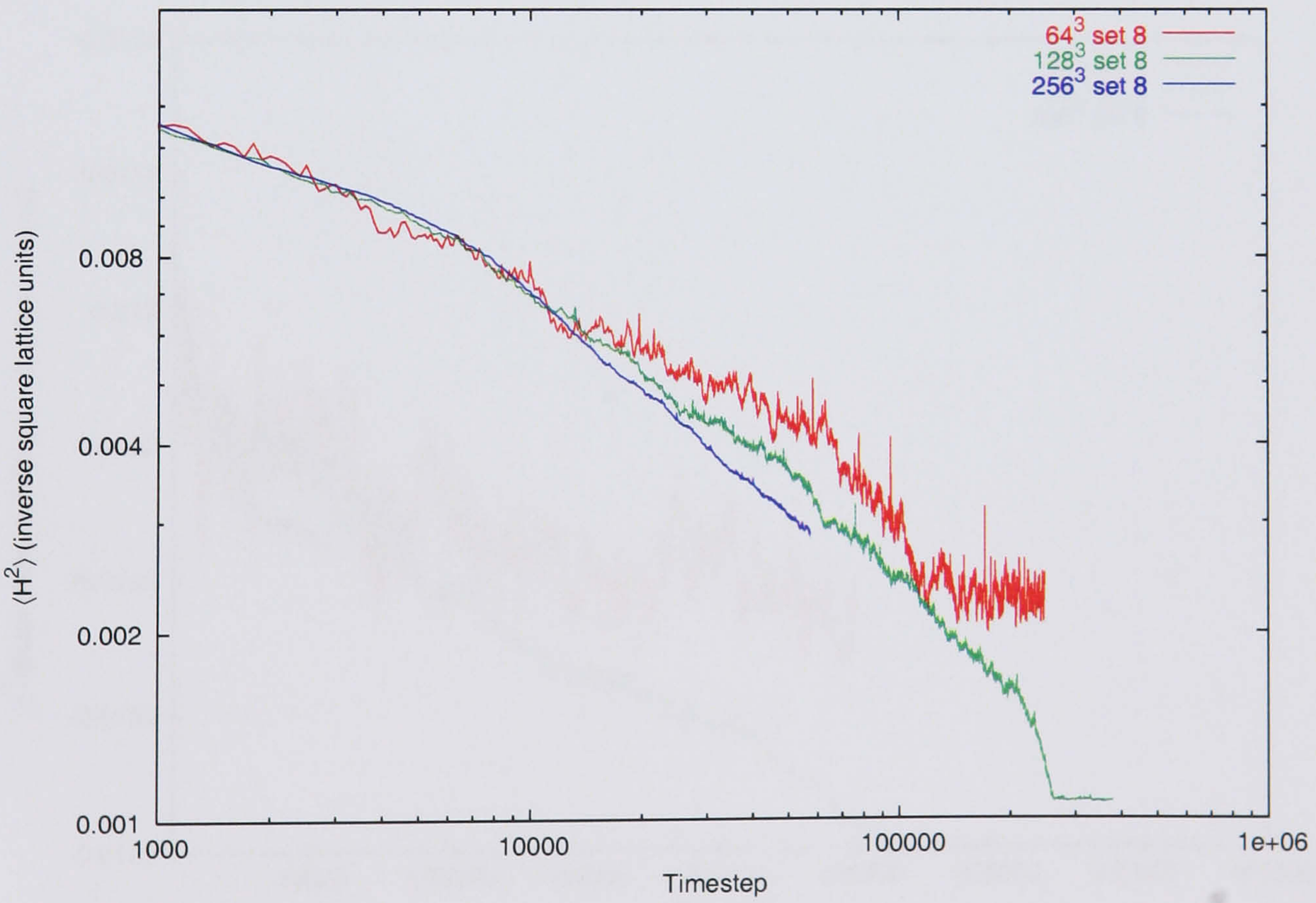


Figure 4.35: Squared mean curvature H^2 averaged over the $\phi = 0$ surface for parameter set 8, showing smaller fluctuations in larger systems, and perfect minimal surface formation in the 128^3 simulation.

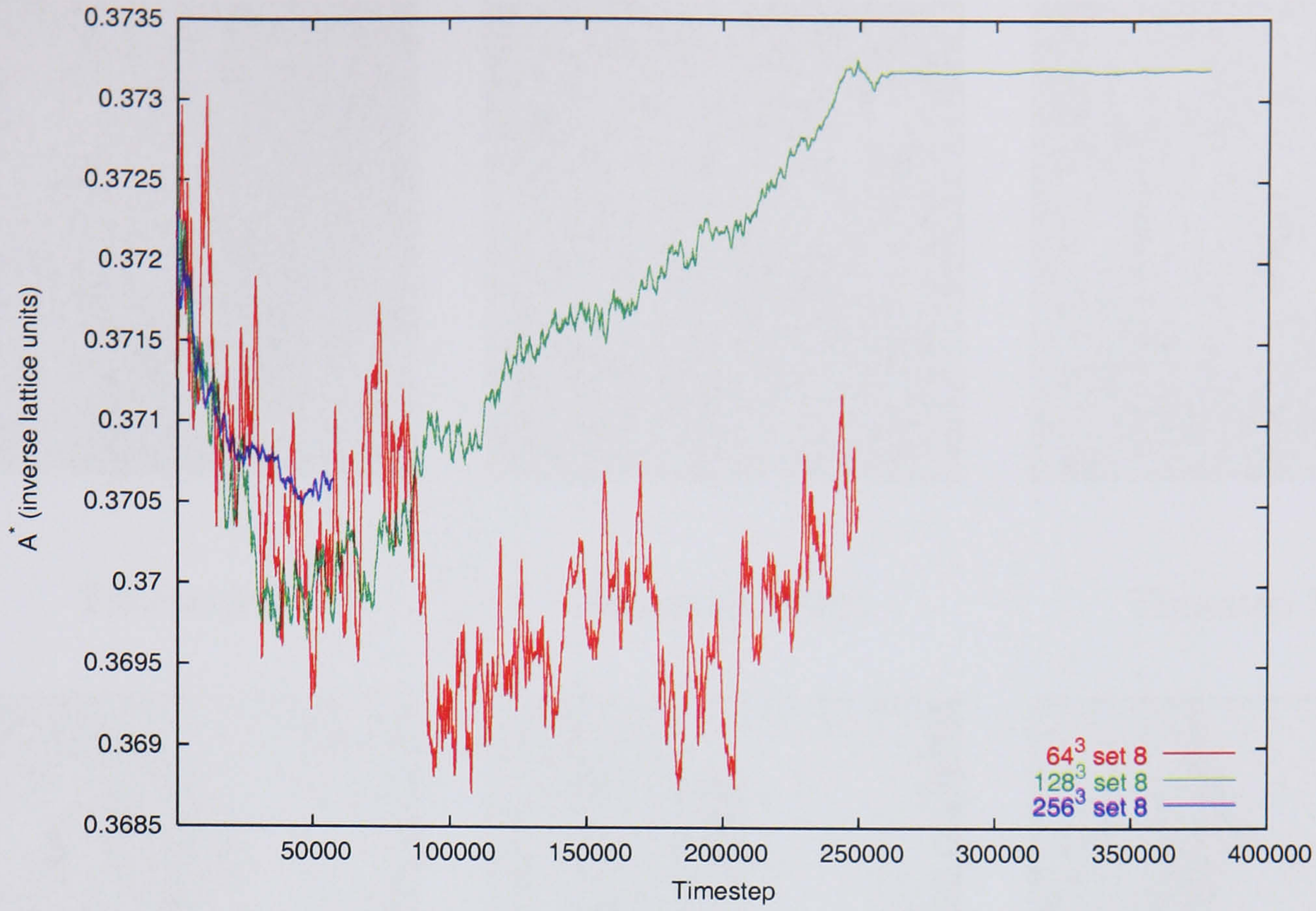


Figure 4.36: Normalized area $A^* = A/V$ of the $\phi = 0$ surface for parameter set 8, for three different simulation sizes. Fluctuations are reduced in larger systems due to more extensive averaging, and finite-size effects are again clear.

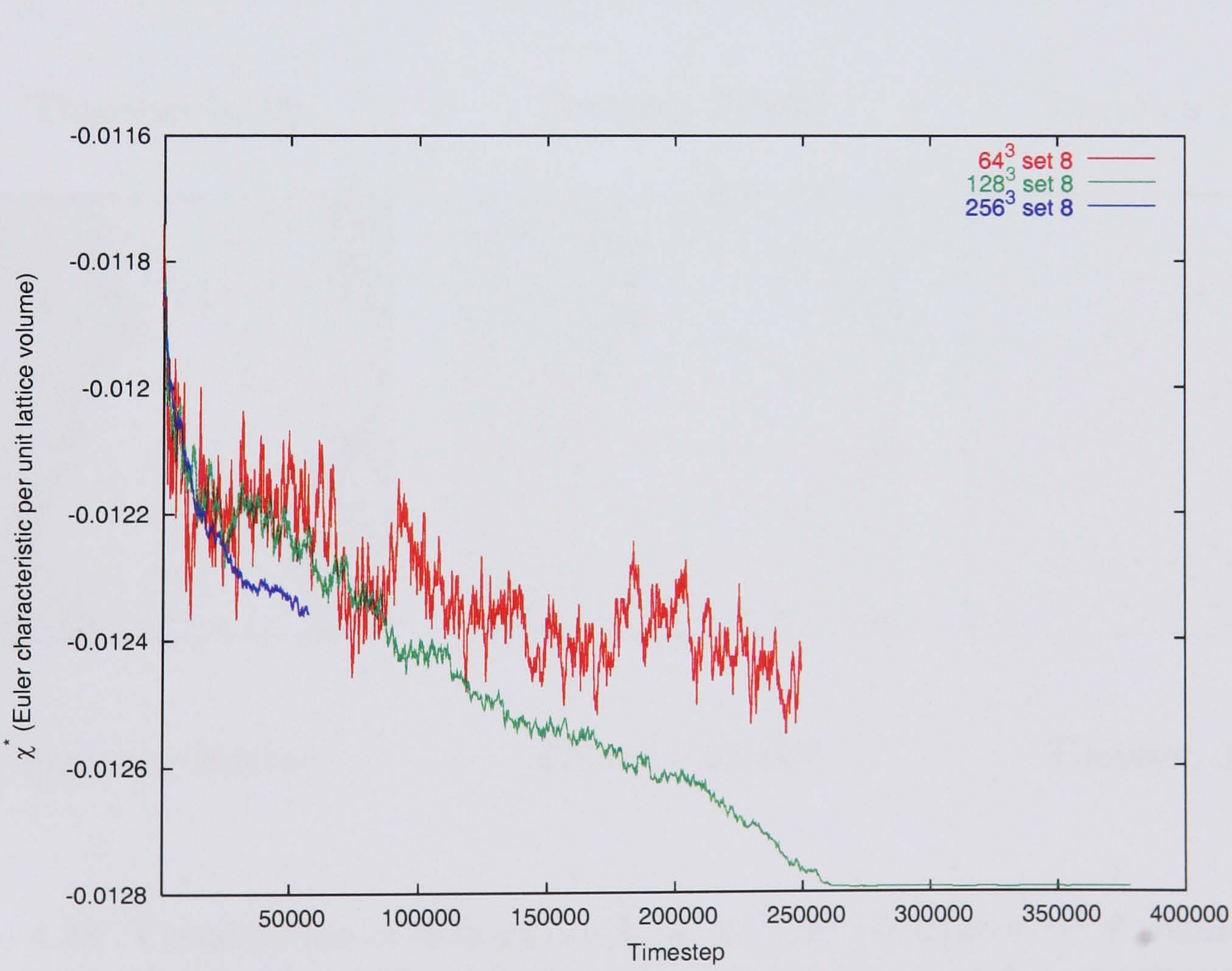


Figure 4.37: Normalized Euler characteristic $\chi^* = \chi/V$ of the $\phi = 0$ surface for parameter set 8.

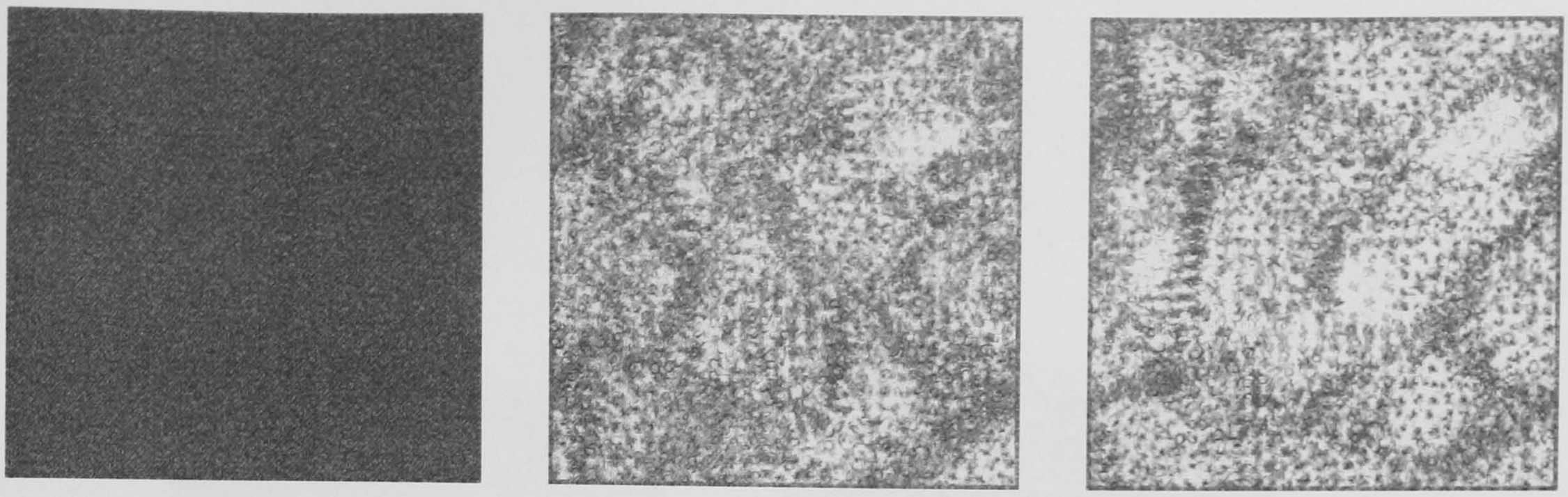
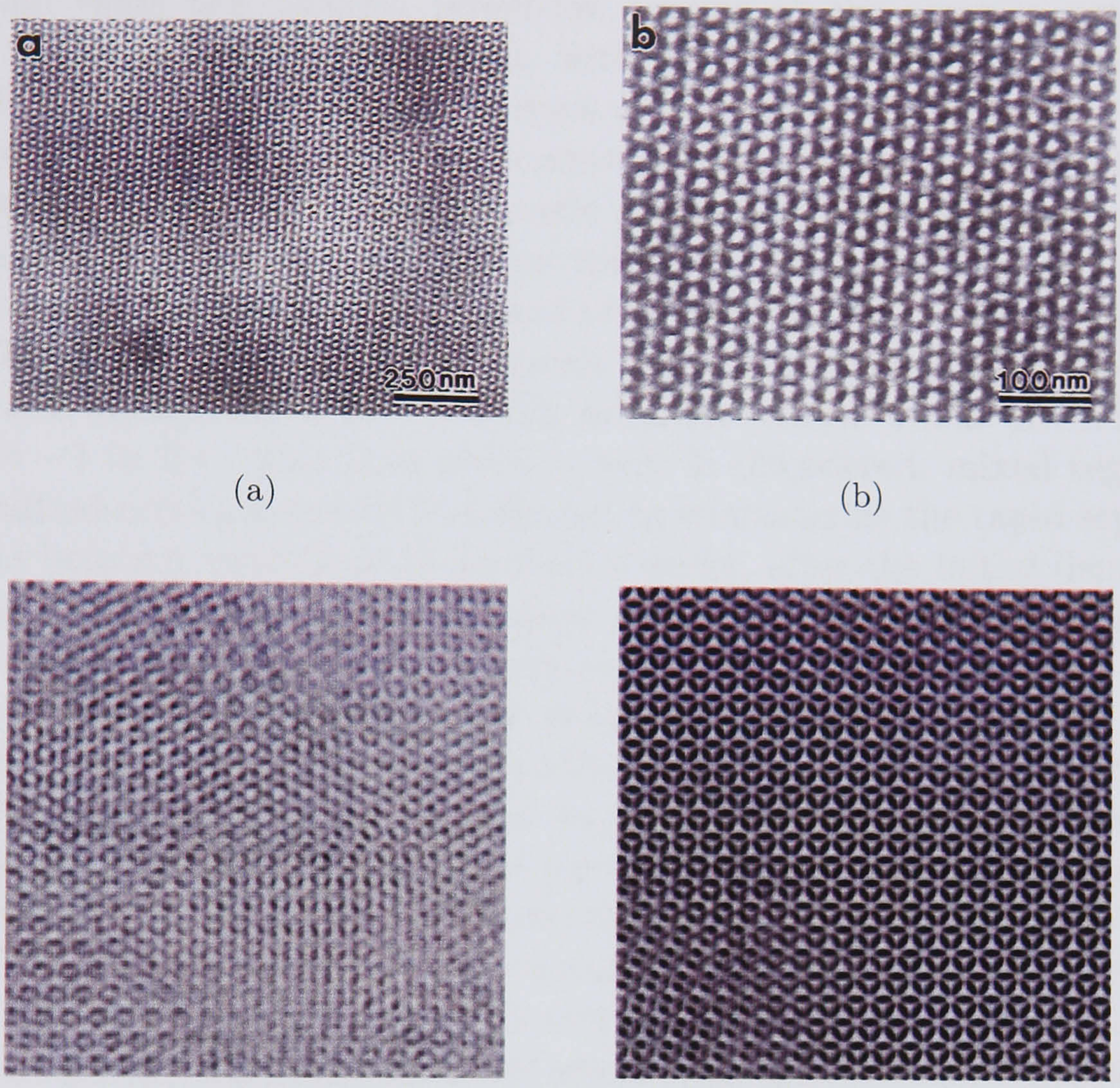


Figure 4.38: Visualization of domain walls in the 128^3 parameter set 8 simulation. Dark regions have a low value of the order parameter $|\phi|$ and a simultaneously low value of $|\nabla\phi|$, and were observed to correspond to domain walls. By timestep 100000, the system contains only two domains; one of these annihilates, leaving an almost perfect gyroid structure by timestep 350000.



(a) (b)

(c) (d)

Figure 4.39: Views down the (111) axis of gyroid mesophases. (a) and (b) show transmission electron micrographs of a triblock copolymer, reproduced from Laurer *et al* [217]. (c) shows a "virtual TEM" orthographic projection volume rendering of part of the 256^3 parameter set 8 dataset at timestep 57500. The significant variation in clarity and density of the wagon wheel pattern is due to the presence of many gyroid domains. (d) shows a virtual TEM of the 128^3 parameter set 8 system at timestep 2×10^5: this system contains two chiral domains, visible as a slight blurring in the top right and bottom left of the image.

4.7 Conclusions

Simulations of the self-assembly of a surfactant gyroid mesophase were performed: for the first time, the simulations were sufficiently large to clearly show the formation and evolution of multiple gyroid domains.

All systems underwent a rapid phase separation, during which oil and water phases formed a network of channels, separated by surfactant. After the phase separation, the systems began to form a gyroid mesophase. At late times, when clear domain walls had formed, power-law scaling of the average squared mean curvature of the system was observed, with exponent close to $-1/2$. This may indicate diffusive scaling of the length scale of the domains themselves.

Analysis of simulated mesophases is nontrivial, due to the difficulty of defining a gyroid order parameter. In a ferromagnetic system, one can use the magnetization $\mathbf{M}(\mathbf{x})$ as an order parameter: all points within a ferromagnetic domain will have the same value of $\mathbf{M}$. In an ordinary fluid phase transition such as spinodal decomposition, the density of one fluid at a point can be subtracted from the density of the other, and normalized to give a scalar order parameter which varies from $+1$ in one fluid to -1 in the other, and which is zero in disordered, mixed regions. This kind of localised order parameter is sufficient to characterize the rapid separation of components before a mesophase is formed; however, after the initial demixing, further ordering of the system is morphological in nature, as gyroid regions are formed. Such a localised order parameter cannot describe this sort of ordering, since it is defined on too small a length scale, and is not morphological in nature.

A set of tools was developed for calculating a polygonal approximation to the $\phi = 0$ surface of a simulated mesophase, in a topologically consistent manner: calculations of useful quantities such as the topologically invariant Euler characteristic, total interfacial area, and interfacially-averaged squared mean curvature were then possible. The average squared mean curvature was observed to vary as a power law with exponent approximately $-1/2$ over the régime where domains are clearly formed but finite-size effects have not yet set in: this may indicate random-walk motion of the domain walls. Direct visualization of the interface shows the existence of chiral domains, as should perhaps be expected for a symmetric amphiphile which has no intrinsic chirality. Once domains effectively disappeared due to reaching the simulation system size, the formation of line dislocations analogous to those in solid crystals was observed.

These simulations have opened up several avenues for further work. Even disregarding the physics involved, the techniques for characterization of lyotropic liquid crystal phases could still be improved, by looking at, for example, localized contributions to the interfacial curvature or Euler characteristic. Gyroid domain walls were observed to have a high density of points where both the local order parameter ϕ and the local order parameter gradient were small: while this may be due to the structure of the $\phi = 0$ surface inside domain walls, an improvement of this technique beyond a simple visual tool could lead to an algorithm for spatial localization of domain walls, and consequently the ability to make more quantitative analyses of the domain behaviour.

The simulations raise a number of physical questions. Those performed so far have all had "oil"/"water" symmetry and a symmetric amphiphile, a situation which may be hard to realize exactly in experiment. The simulations produced chiral domains: it would be interesting to see if these could ever be produced experimentally.

since the existence of such domains would affect the feasibility of producing gyroidal chirally selective filters. Observations of gyroid chirality would be difficult to make using TEM or SAXS techniques, but might be possible using, for example, transmission electron microtomography[217].

Finally, it is interesting to note that the lattice Boltzmann method allowed the bridging of two different length scales in these simulations: the scale ($\sim$ 5 lattice sites) of the single-fluid-component domains, and the scale ($\gtrsim$ 50 lattice sites) of gyroid domains.

Chapter 5

Conclusions

The Shan-Chen lattice Boltzmann Equation algorithm has been implemented in both two and three-dimensional versions, with the latter being capable of modelling the dynamics of surfactant mixtures. Both of these implementations have been demonstrated to have qualitative and quantitative agreement with existing theory, experimental observations, and other numerical methods.

Despite its basis in a "bottom-up" particle interaction scheme rather than any sort of free energy argument, the Shan-Chen model showed agreement with a Cahn-Hilliard model of early-time spinodal decomposition. The $\frac{2}{3}$ scaling exponent corresponding to hydrodynamic phase separation was observed, but the viscous scaling régime was not observed. It remains possible that this régime could be found through a careful consideration of the parameter space of the model. Dynamical scaling was satisfied where expected, but it was also observed to break down in certain régimes, producing morphologies similar to those observed experimentally.

An improved lattice Boltzmann model for Hele-Shaw flow was developed, showing improved accuracy and stability. This could form the basis for much future work, including closer examination of the Saffman-Taylor instability.

Finally, some extremely large scale calculations were performed in order to produce the first dynamical simulations of minimal surface surfactant mesophase domain growth. Novel techniques for analysing these calculations were investigated and used, suggesting scaling laws for domain growth, demonstrating the presence of chiral domains in the gyroid mesophase.

The field of lattice Boltzmann modelling has moved on considerably since this work was commenced. In particular, the issue of numerical stability has been tackled with considerable success: what were once described in the literature as "ill-understood" numerical instabilities[228] are now generally seen as a consequence of the lack of H-theorem inherent in the lattice BGK model, and stabilised BGK-like models have been constructed[228, 229] and are coming into greater use[230, 136]. Improved stability is also available through generalization of the BGK model to use multiple relaxation times[65].

However, even the entropically stabilised models are only for single-component fluids. Extending this stabilisation to the multicomponent interacting case is extremely difficult: any kind of entropy-like functional which covers the case of interacting fluids will typically contain terms nonlinear in the single-particle distribution function: this means that it may vary over the advection step, and that it is much harder to extremize. In fact, as has been known for some time[67], fluid interactions are an intrinsically two- (or more-) body problem, so a description at the single-

particle distribution function level may not be sufficient. More advanced models could perhaps be obtained by moving further up the BBGKY hierarchy: either by producing a discretized "lattice BBGKY" model, analogous to the evolution equation of the continuum two-particle distribution function in the same manner that lattice Boltzmann is analogous to the evolution of the continuum single-particle distribution function, or by studying behaviour higher up the BBGKY hierarchy in order to incorporate many-body effects into the Boltzmann collision operator.

The number of application domains of lattice Boltzmann has continued to grow, with an increasing number of practical uses in industrial situations, particularly for single-component flow modelling, where it continues to vie with conventional CFD at high Reynolds numbers. However, the complex fluid lattice Boltzmann models tend to lack any kind of molecular specificity: one does not simulate a particular fluid such as water or oil, but match up dimensionless numbers to the scenario to be modelled. While there can be an explicit connection to macroscopic free energies, there is rather less contact with more molecular-level models, and this could form a fruitful and useful basis for future work.

Appendix A

Lattice Boltzmann Implementation

The lattice Boltzmann calculations presented in this thesis were performed using two programs, called LB2D and LB3D, both of which have grown into multipurpose simulation codes used by several other individuals and groups.

A.1 LB2D

LB2D is a two-dimensional implementation of the Shan-Chen lattice Boltzmann algorithm[57, 231] for simulating mixtures of interacting fluids, and has been used for studies of spinodal decomposition[191], viscous fingering[108, 106], and flow of non-Newtonian fluids in porous media[232], as well as being used during some investigations into computational steering[71].

It supports the following features:

- Two component fluid simulation
- Simulation of flow in porous media
- Hele-Shaw boundary conditions
- Simulation of power-law non-Newtonian fluids
- Taskfarmed simulations
- Scripting using the Perl language

In LB2D, the complete state of a single lattice Boltzmann simulation is held in an object called the `sim` structure: the entire program design revolves around operations on this structure. It contains the array of lattice sites holding the fluid state, as well as the simulation parameters. These parameters may be divided into two categories: parameters which are static and unchanging, such as the dimensions of the simulation lattice or parameters describing the initial state of the system, and parameters which could conceivably be changed during the course of the simulation, such as coupling constants and forcing rates. The code is then structured as a set of methods which act upon the data in this structure. Global variables are avoided, since they tend to hinder maintenance and reusability of the code, and could also cause problems were threaded parallelism ever implemented.

A.2 LB3D

LB3D is a three-dimensional implementation of the amphiphilic lattice Boltzmann model of Chen Boghosian, and Coveney[68]. It can simulate the flow of ternary amphiphilic fluids in porous media, and has been used for investigations of the lamellar and cubic P surfactant phases[180], phase separation[31, 233, 234], complex fluids in porous media and under shear[235], spinodal decomposition[181], and the gyroid mesophase[182, 183]. It was designed from the start to run on distributed-memory parallel processing architectures, and has been placed in the top scalability class for parallel codes running on the HPCx supercomputer.

It supports the following features:

- Simulation of three-component fluids (of which one is amphiphilic)
- Simulation of flow in porous media
- Lees-Edwards sheared boundary conditions
- Remotely steerable via the RealityGrid steering interface
- Parallel processing via MPI and domain decomposition
- Taskfarmed simulations
- Parallel IO using the HDF5 library

LB3D is similar in spirit to the ME3D code for ternary amphiphilic lattice gas simulations[13, 227]; both codes were used for a comparison of the lattice gas and lattice Boltzmann methods in [31].

A.3 Data structures

At a given timestep, the complete state of a lattice Boltzmann fluid mixture is given by the distribution function $f_i^\sigma(\mathbf{x})$, over all sites $\mathbf{x}$, lattice vectors i, and fluid components σ. The most basic data structure in LB2D and LB3D describes the state of a single lattice site, containing a double-precision floating point number for each component of the discrete distribution function of each of the two components, plus an extra bit of information to describe whether or not the lattice site is occupied by an obstacle. The state of the system is then composed of a large array of these data structures, one for each lattice site.

It should be noted that this way of storing the fluid state is not particularly well suited to simulations of flow through porous media, since the amount of memory allocated is proportional to the number of lattice sites, but the amount of memory actually used is proportional to the number of pore-space sites. Techniques for running lattice Boltzmann simulations on sparse grids containing only the pore-space sites are described in the literature[236], but have not been implemented in LB2D or LB3D at present. Nonetheless, it has been possible to perform large-scale porous media flow calculations with LB3D[235].

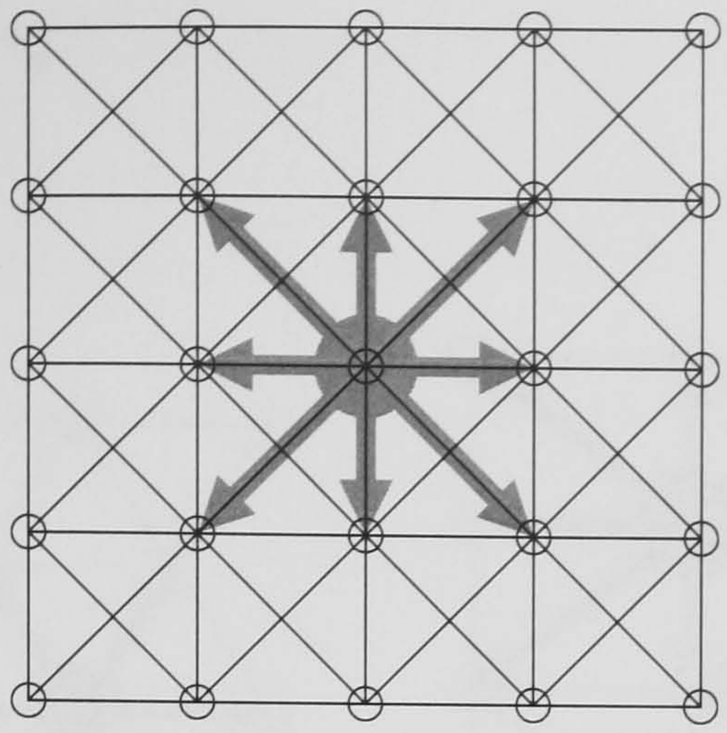


Figure A.1: A set of 5×5 sites on a D2Q9 lattice, with the lattice vectors on the central site highlighted. The sites sit on a Cartesian lattice.

A.4 Choice of lattice

The choice of lattice in an LB calculation is governed primarily by the physics involved, since only a certain set of lattices have the symmetry properties required to give the correct hydrodynamic behaviour. However, these constraints still do not unambiguously determine which lattice should be used in all cases, particularly for athermal simulations.

For two-dimensional athermal simulations, the D2Q6 or "FHP" lattice could be used, as could the D2Q7 lattice (indentical to D2Q6 but with a zero-velocity vector as well), or the D2Q9 lattice. LB2D uses the nine-velocity D2Q9 lattice (figure A.1): two-dimensional lattice Boltzmann simulations on this lattice are more numerically stable[237], and it also has the convenient feature that the sites sit exactly on a Cartesian lattice, with additional lattice vectors running in the diagonal directions. This makes implementation of the algorithm particularly simple: for the D2Q7 lattice, adjacent rows of sites are stored slightly differently in memory, whereas for D2Q9, all lattice sites can be treated identically (Figure A.2).

LB3D was designed to be as similar as possible to the ME3D lattice gas code, which used a 19-velocity lattice with degeneracies to give 26 distinct lattice vectors. LB3D therefore used a 19-velocity lattice with weighting factors to represent the degeneracies and ensure isotropic hydrodynamics, as shown in figure A.3. The lattice sites for D3Q19 (and indeed the slightly simpler D3Q15) lie on a cubic Cartesian grid, which again simplifies implementation.

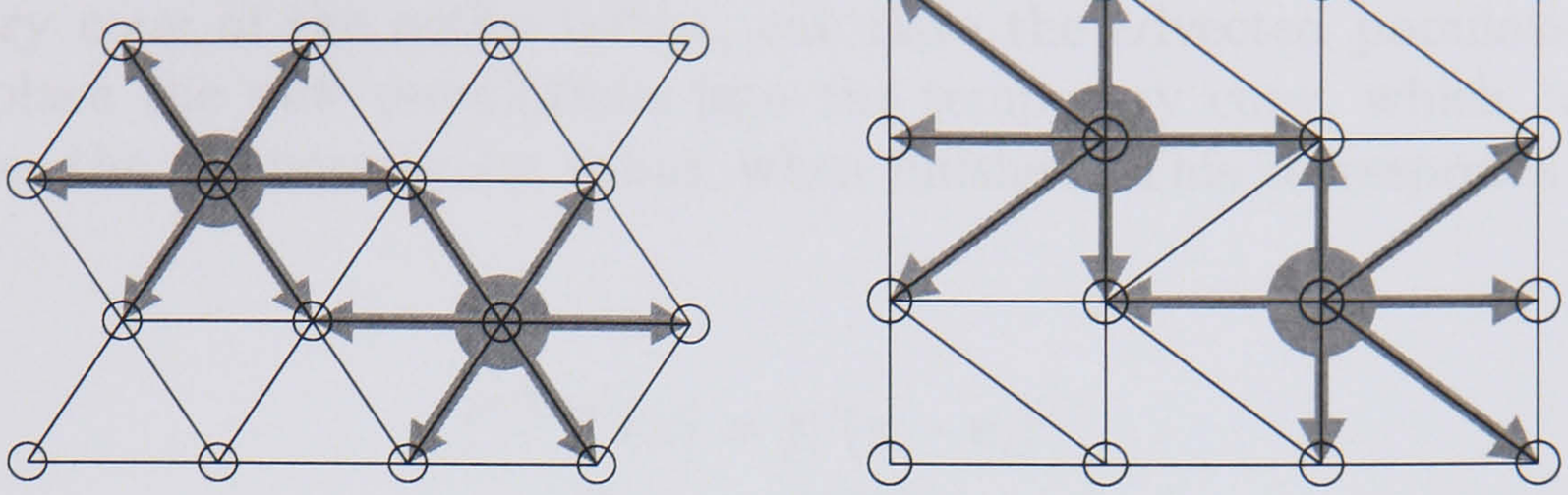
A.5 The moving buffer algorithm

The advection step maps the fluid state f_i^σ into a new state $f_i^{\sigma(\text{new})}$, by the operation

$$f_i^{\sigma(\text{new})}(\mathbf{x} + \mathbf{c}_i) = f_i^\sigma(\mathbf{x}) \tag{A.1}$$

operating at all sites $\mathbf{x}$. The operation described in equation A.1 looks at lattice site $\mathbf{x}$, and runs through each of its lattice vectors, moving particles on that site with velocity $\mathbf{c}_i$ to their new location at $\mathbf{x} + \mathbf{c}_i$. For implementation purposes, it can be easier to think of this operation in the opposite sense,

$$f_i^{\sigma(\text{new})}(\mathbf{x}) = f_i^\sigma(\mathbf{x} - \mathbf{c}_i), \tag{A.2}$$



(a) (b)

Figure A.2: (a) Sites on a D2Q6 or D2Q7 lattice, with the lattice vectors on two sites highlighted. (b) D2Q6/D2Q7 sites mapped onto a Cartesian lattice. Suppose that these sites are stored consecutively in memory, left to right and then top to bottom: the memory-space relationship between adjacent lattice sites is then different for alternate rows.

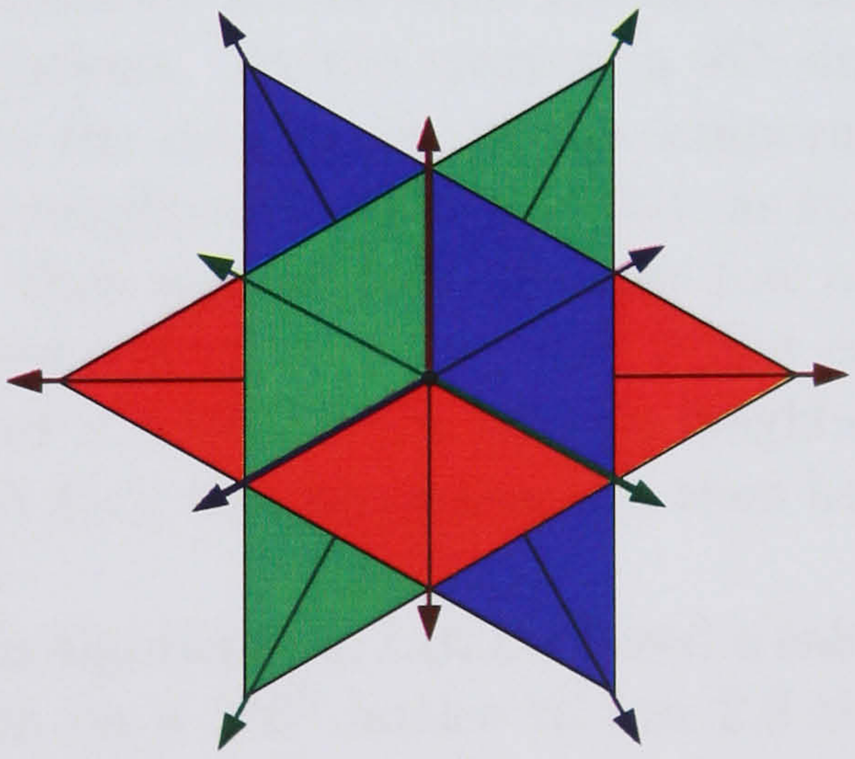


Figure A.3: The D3Q19 or D3Q26 lattice. If three perpendicular unit squares intersect at the origin, then the lattice vectors are composed of the diagonals and normals of each square, plus a rest vector. In the D3Q26 lattice, the square normal vectors have twofold degeneracy, effectively giving 26 vectors from the original 19.

representing all of the particle populations at the site $\mathbf{x}$ being replaced by their advected values from adjacent sites.

Unfortunately, the advection step cannot be performed "in place", since advection of particles into a lattice site requires knowledge of the distribution function at adjacent sites. The easiest way to implement the advection step is to allocate a temporary copy of the entire lattice, calculate the advected populations site by site, and place the new populations into the temporary copy, which can then be written over the old population values when finished. This corresponds to the two operations

$$f_i^{\sigma(\mathrm{tmp})}(\mathbf{x}) = f_i^{\sigma}(\mathbf{x} - \mathbf{c}_i) \tag{A.3}$$

$$f_i^{\sigma}(\mathbf{x}) = f_i^{\sigma(\mathrm{tmp})}(\mathbf{x}). \tag{A.4}$$

If the temporary copy of the lattice is persistent, then the second operation can be accomplished without any memory transfer, by swapping pointers to the lattice and its temporary copy.

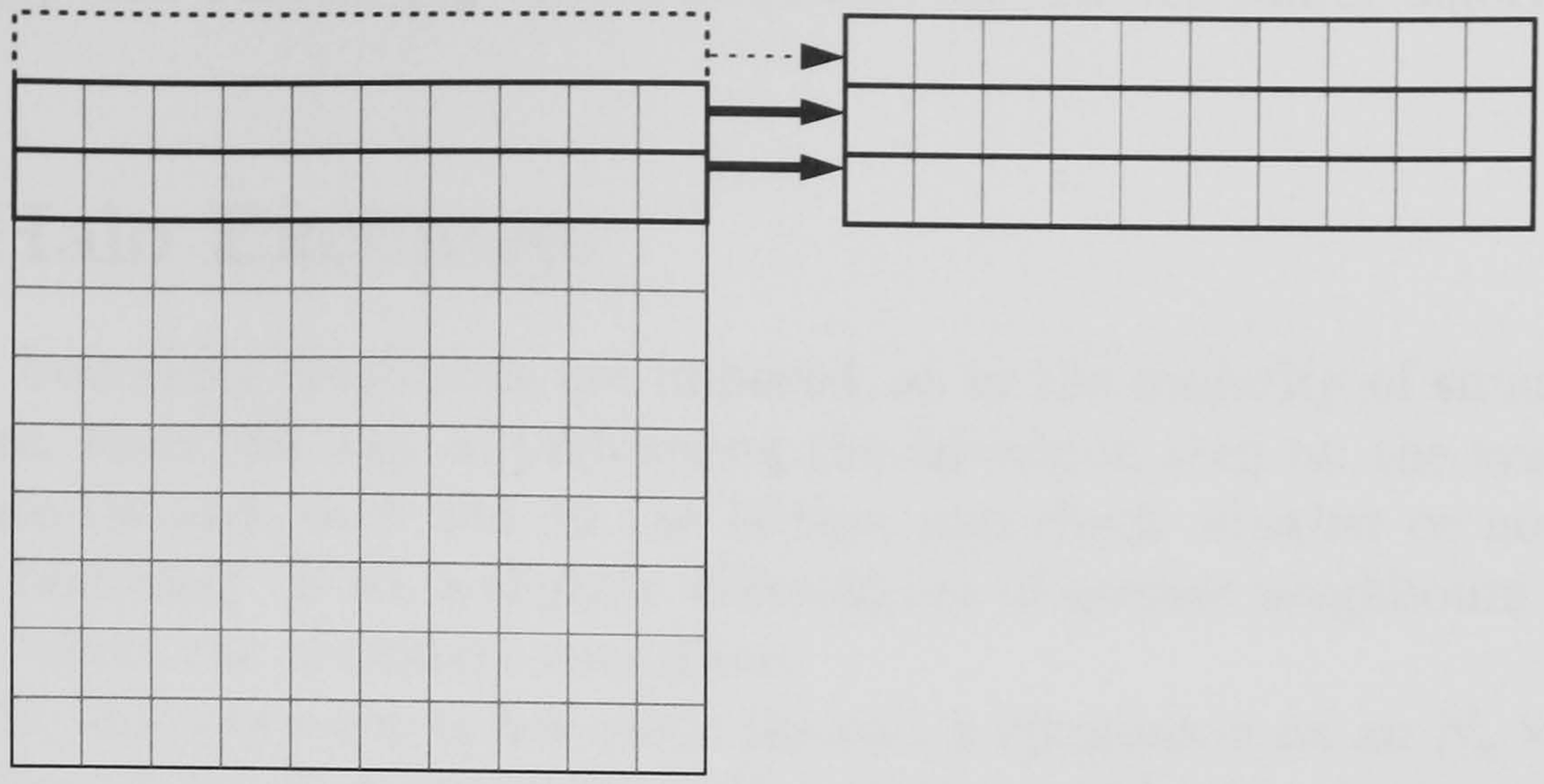
A drawback of this procedure is that it effectively doubles the memory requirements of the simulation. This is acceptable for two-dimensional simulations, where a 1024^2 D2Q9 lattice for a single-component simulation would contain around 1.89×10^7 floating-point values, taking up around 151MB of memory at double precision, well within the capability of a modern computer. However, a 256^3 simulation on the D3Q19 lattice would take up 2.55GB of memory without the temporary array: including the temporary array doubles this to 5.1GB, which is more than the address space of a 32-bit machine, and a ternary amphiphilic simulation more than triples these memory requirements.

As noted by Martys and Hagedorn[238], and implemented by Kevin Roy of Manchester Computing in LB3D, the temporary array does not need to be the size of the entire lattice. Instead, a temporary circular buffer can be used, which simply contains three planes of lattice sites for 3D simulations, or three rows of lattice sites for 2D simulations. At the start of a 2D simulation, the first row of lattice sites is copied into the middle row of the temporary buffer; the other rows are filled with the nearest neighbours of the first row, as shown in figure A.4(a). The new advected values are then written into the first row of the lattice, as shown in figure A.4(b). The topmost row of the temporary buffer can now be discarded, and the buffer updated so that it contains the nearest neighbours of the second row of lattice sites, as in figure A.4(c); this procedure can then be repeated until the entire lattice has been updated.

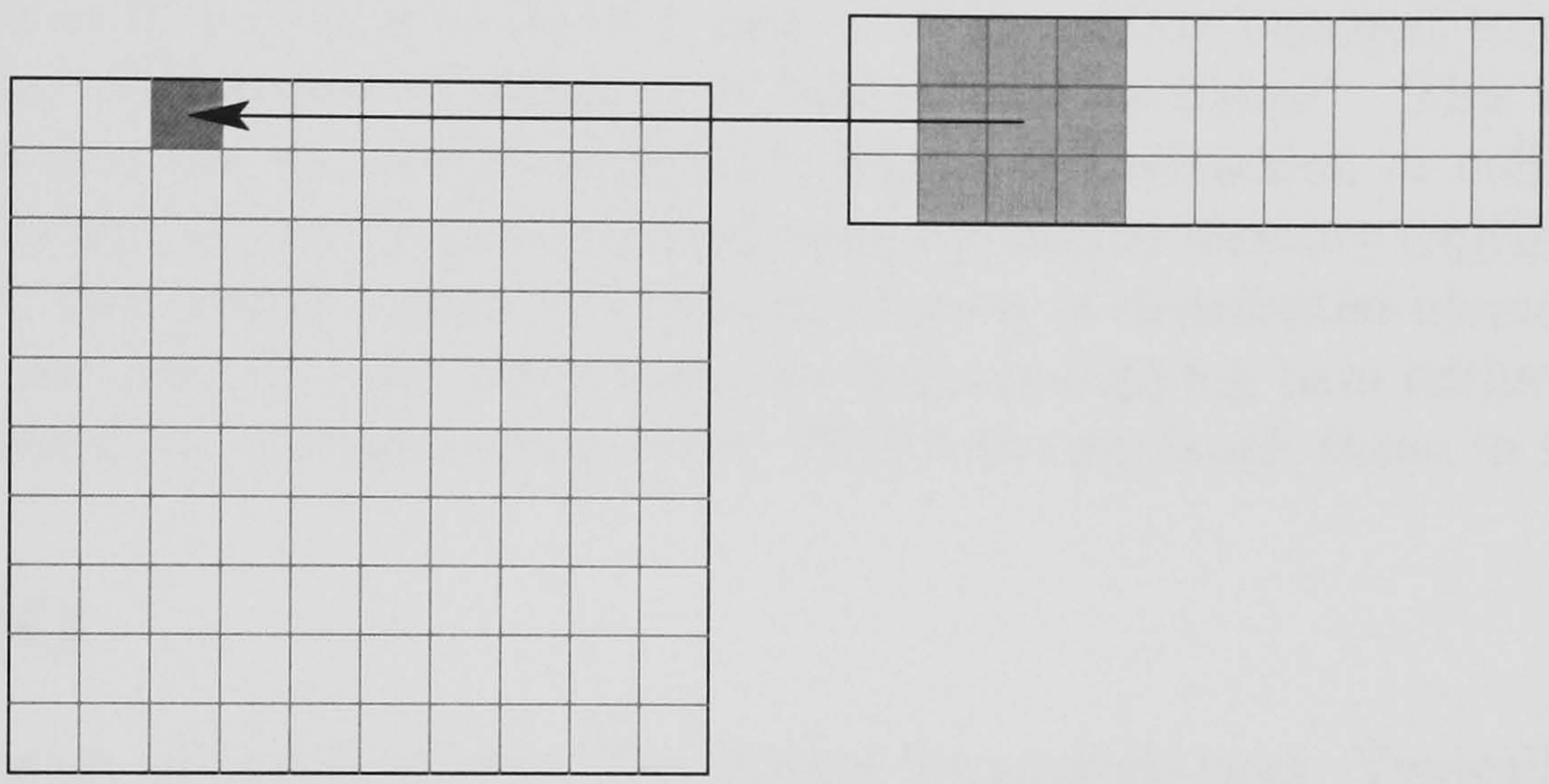
Implementation of this algorithm in LB2D caused a calculation of 1000 timesteps of spinodal decomposition on a 256^2 lattice to run 2.6 times faster, as well as reducing the memory requirements. This is unsurprising, since the performance of LB codes tends to be limited by memory bandwidth rather than floating-point performance[239, 65].

Bounce-back boundary conditions can be implemented in the advection loop, by reflecting particles off obstacles before the advection step itself.

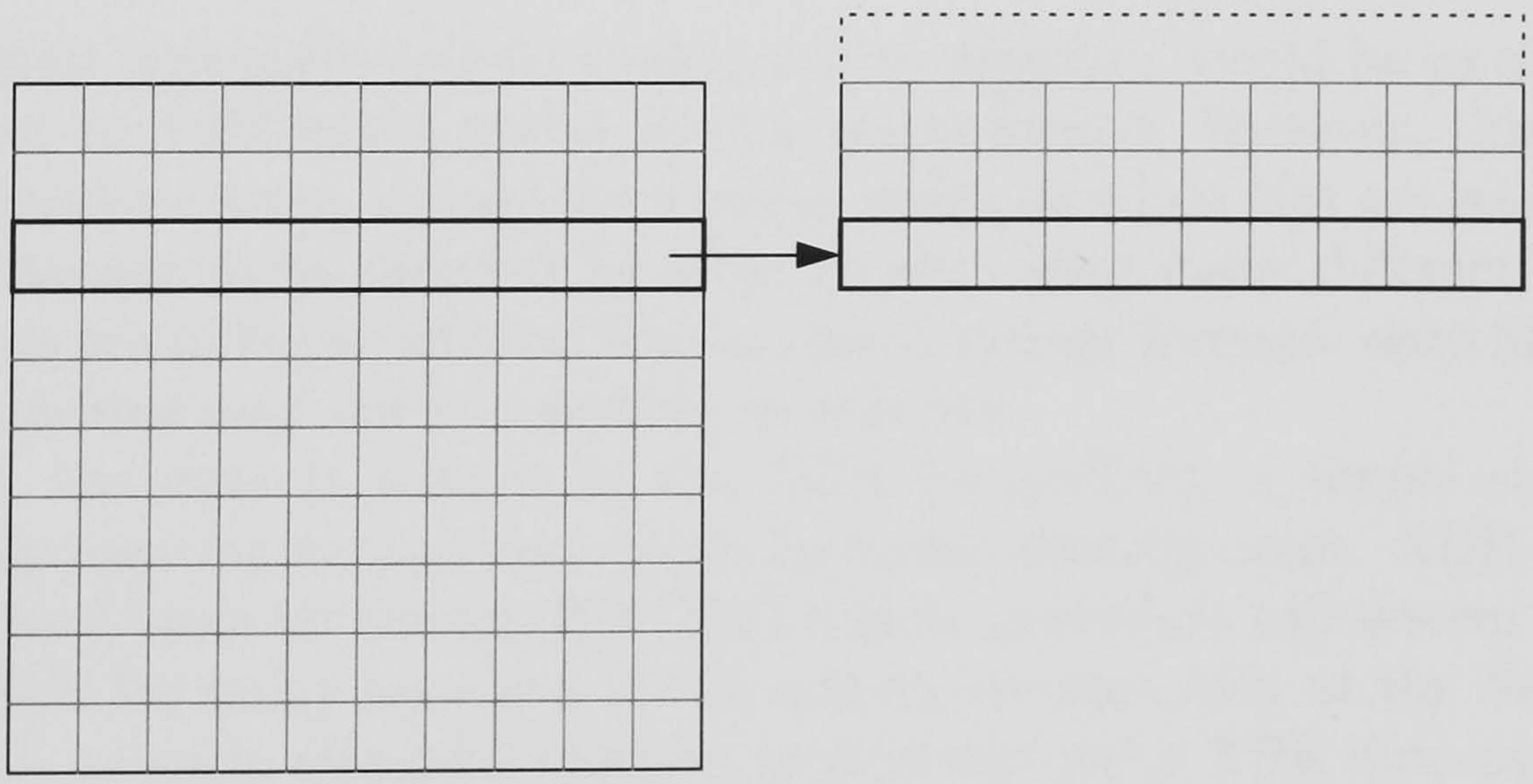
In a single-component lattice Boltzmann algorithm, the collision step is local: the new state of each lattice site does not depend on the state of its nearest neighbours. This is not the case for interacting LB algorithms, where the forcing terms require calculation of gradients. However, these calculations only depend on the (collision-invariant) total density of each component at adjacent lattice sites, so that the



(a) The initial populations of the nearest neighbours of the first row of lattice sites are copied into the moving buffer.



(b) The first row of lattice sites is updated with advected populations, from the nearest-neighbour data kept in the moving buffer.



(c) The top row of the moving buffer is discarded, and another row loaded into it, so that it now contains the original nearest-neighbour data for the second row of lattice sites.

Figure A.4: Use of a moving circular buffer reduces the memory requirements of the advection step.

collision step can still be performed in-place: the moving-buffer algorithm is not required.

A.6 Halo Exchange

If periodic boundary conditions are imposed, as in the majority of simulations described here, then one way of performing the advection step on the system would be to iterate through each site on the lattice, and check whether or not it lies on the lattice boundary. If so, a slightly different set of nearest neighbours could then be used, to effect the boundary conditions.

In LB2D, this approach is not used. Instead, a simulation on an $N_x \times N_y$ lattice actually takes place on an $(N_x + 2) \times (N_y + 2)$ array of lattice site structures, with the $x = 0$ column a copy of the $x = N_x$ column, and the $x = (N_x + 1)$ column a copy of the $x = 1$ column. The advection and collision steps only operate on the sites with $1 \leq x \leq N_x$ and $1 \leq y \leq N_y$. After one of these steps, the "halo" regions at $x = 0$, $y = 0$, $x = N_x + 1$, and $y = N_y + 1$ are changed to contain the correct data: the process of doing so is called "halo exchange". This means that each lattice site can be treated identically during the advection or collision steps, and the halo regions can be implemented using optimized memory copying routines. In addition, this design makes the implementation of distributed-memory parallel processing particularly easy, since then the memory-copying halo exchange code is simply replaced by communication code. This is the approach taken in LB3D.

A.7 IO

Any simulation will need to write files of data for analysis later. Typically, these do not need to contain the state of the entire system, but merely a subset, such as the density or velocity of a given component at each lattice site, corresponding to one or two floating-point numbers per site, rather than eighteen or more for the complete state.

The fastest approach, and the easiest one to program, would be to simply write the data out to a file in the native floating-point format. However, this can cause awkward problems later, during the analysis stage, or when files are retrieved from long-term storage to be decoded by someone else, since many different computing architectures use different internal floating-point storage formats: data files dumped from one machine may not be readable on another.

Instead, the data is written in the XDR format[240], a simple standard for portably representing various data types including floating-point. XDR forms part of the low-level basis for the SunRPC[241] remote procedure call system which is in turn the basis for many standard UNIX system services such as the Network File System[242]: as such, standard routines to read and write XDR data can be found on any UNIX system, and many UNIX-like systems as well.

In addition to routines to dump the density field to a named file, routines also exist to copy it into a newly-allocated memory buffer. As well as being of use in implementing the file IO code, such routines allow data to be passed in a useful format to other pieces of code linked with LB2D, for example, for conversion into images.

Operating System	Hardware
Linux, FreeBSD, Microsoft Windows (under Cygwin)	IA32
Linux	IA64 , X86-64
Linux, Solaris 9	Ultrasparc I & II
IRIX 6.5	SGI Origin (MIPS R12000)
MacOS X	Apple G4 and G5 machines
AIX	IBM SP3 and SP4
Tru64	Alpha

Table A.1: Platforms on which LB2D has been built and run

A.8 Portability

A.8.1 LB2D

An important design goal of LB2D was portability: the ability to build and run the code on a wide range of platforms. To achieve this, the code was written entirely in strict ANSI C: while some compilers support "extensions" to the language, use of such extensions would mean that the code would not build with any other compiler. The adherence to the ANSI standard[243] was verified by using the `-Wall -pedantic` options to the GNU C compiler. Use of portable data formats as detailed above ensured that simulation data could be exchanged across platforms.

It was ensured that the core code, which could read and parse ASCII input files and then perform the corresponding simulation, would compile without requiring access to any software libraries beyond those which must be supported by an ANSI C compiler running under a UNIX environment supporting SunRPC. The code supports the use of external libraries: in addition to various graphics libraries for writing data directly into viewable image files, the MPI library[244] can be used for parallel taskfarmed simulations, the FLXmitter[245] library can be used to send data from running simulations over a multicast network, and all of the LB2D functionality can be driven through a programmable layer as described in section A.12.2. The presence of these libraries is detected by a configuration shell script, generated by the GNU Autoconf[246] package. The configuration script also allows certain features of the code to be turned off at compile-time, in order to improve performance.

As a result of these measures, the code has been built with ease on every UNIX or UNIX-like platform available to the author, most of which are listed in table A.1. The code has been designed to build on all of these platforms without requiring any changes or manual configuration first, and has been observed to build in this way on entirely new platforms without requiring any adjustment.

A.8.2 LB3D

A goal imposed on LB3D at the start of its development was that its design and operation should remain as similar as possible to the older ME3D code. LB3D gradually evolved away from this approach as new methodologies were explored, and shortcomings of the existing design were found.

Because of this initial requirement, LB3D was written in the Fortran 90 language. Fortran is the de facto standard language for high-performance numerical codes: this is generally attributed to the availability of high-quality Fortran numerical libraries, the large corpus of existing Fortran code which must be maintained or with which

interoperation is desirable, and certain features of the language. such as a first class complex number type, the relative lack of aliasing issues, and the extremely powerful array syntax of Fortran 90.

However, the choice of Fortran had its drawbacks. Support for certain language features (such as recursion, short-circuit evaluation, or the Fortran 95 `forall` construct) varies across implementations, so that portable Fortran codes must be written in a restricted, sometimes awkward, and often idiosyncratic[247] subset of the language.

IO operations in Fortran can be cumbersome, since the language does not adopt the now rather common view of files as streams of bytes, each comprising eight bits. The on-disk format of Fortran "unformatted" binary files is implementation-dependent, making exchange of such files with programs not written in Fortran rather difficult. A UNIX environment compliant with the Single UNIX Specification [248] provides the C programmer with an unsigned eight-bit byte datatype and functions to manipulate files at the byte level: at the very minimum, more complicated operations can be built out of these elements. Unfortunately, while it is usually possible to perform such operations from Fortran, the way in which this is done can vary widely between implementations, and can often be cumbersome. The semantics of the `NAMELIST` format also differ between Fortran implementations, with the result that namelist files written on one machine can cause program crashes when read on another. Finally, there is no standard Fortran support for elementary filesystem operations such as the creation or deletion of directories. nor is there any standard Fortran interface to the standard UNIX API. This situation echoes Kernighan's criticism of the Pascal language[249], that attempts to provide missing features "add to the utility of the language for one group, but destroy its portability to others".

Some of the shortcomings of Fortran can be solved by linking Fortran programs with libraries written in C. Unfortunately, there is no Application Binary Interface (ABI) for Fortran, and in particular, there is no standard mapping from Fortran identifiers to the corresponding symbol names in object files. This means that a subroutine called `foo` in a Fortran program may be assigned the symbol name `_FOO` on one platform, `FOO` on another, or `foo`, `FOO_`, or `_foo` on others. Worse, on a single platform, the symbol naming convention can vary according to how the compiler is invoked. Platform-dependent features in C can, to an extent, be dealt with quite easily through the C preprocessor, but there is no corresponding feature in Fortran, and while many compilers offer the option of preprocessing source code, the ways in which they do so and are told to do so also vary.

In LB3D, these shortcomings were dealt with in a variety of ways, each trading portability and standards compliance with performance and ease of implementation.

The most portable solution, but also the slowest, is to write output files in native Fortran binary format, and then invoke a Fortran postprocessing program after the simulation has finished to write the data in ASCII format. The ASCII data is piped into a C program which can then save the data in a portable format such as XDR. However, this procedure can be very slow due to the large amounts of ASCII data involved, and small inaccuracies can creep into the data through the ASCII conversion process.

Other solutions all involve linking Fortran with C libaries. While this is impossible to do in the most general case due to the lack of any standards for doing so. in practice the number of naming conventions in use is quite small, and it is usually

Operating System	Hardware
Linux	IA32
Linux	IA64 , X86-64
IRIX 6.5	SGI Origin
AIX	IBM SP4
Tru64	Alpha

Table A.2: Platforms on which LB3D has been built and run

possible for a script to determine which convention is in use, and alter the C code accordingly. However, any time a completely new convention is encountered, the build system must still be adjusted. Jens Harting linked LB3D with Frans van Hoesel's XDRF[250], which allowed XDR data to be written from Fortran. Collaborators at the Edinburgh Parallel Computing Centre linked LB3D with the HDF5[251] library. This library gives an interface to a portable file format, and can also take advantage of the MPI2 parallel IO system, if available, to provide faster data transfer on some platforms. However, the HDF5 library itself tended to require manual patching before it would build on many platforms.

The platforms on which LB3D has been deployed are listed in table A.2. Porting to new platforms tended to require significantly more work than was the case with the simpler (and more aggressively portable) LB2D code.

A.9 Testing

LB2D and LB3D were tested at different levels of granularity during their development: this helped to find and locate bugs quickly, and to prevent bugs from being introduced. Several different testing styles were used as the programs were developed.

A.9.1 Unit tests

A unit test[252] is a highly localised test, designed to examine the behaviour of a small part (usually a single function) of the application under development, and verify that it conforms to the expected behaviour. These would often take the form of "sanity checks", ensuring that, for example, total mass and momentum are conserved under the advection, collision, and halo exchange operations; that total mass of each component is conserved individually, and that functions to access and modify the particle populations work as expected.

A.9.2 Functional tests

A functional test is a larger-scale test of the functionality of a complete program or system. For LB2D and LB3D, this is accomplished through simulation of simple hydrodynamic systems for which an analytical solution for the velocity field is known. It should be noted that these tests were designed as a development aid to ensure that the code produced reasonable results, rather than as stringent tests of accuracy, which were typically performed manually.

Poiseuille flow

A fluid of kinematic viscosity ν in a two-dimensional channel of width L subject to a body force of magnitude G will have an equilibrium velocity profile of the form

$$v(x) = \frac{G}{2\nu}\left(\frac{L^2}{4} - x^2\right) \tag{A.5}$$

It can be verified automatically that the flow profile is quadratic, and has the correct velocity peak $v_{\max} = GL^2/(8\nu)$.

Shear wave decay

Consider a system with periodic boundary conditions, filled with fluid of kinematic viscosity ν, with an initial velocity field of the form $v_y = v_0 \sin kx$ for some wave vector k. The appropriate solution of the Navier-Stokes equation is

$$v_y(x,t) = v_0 e^{-\nu k^2 t} \sin kx \tag{A.6}$$

The amplitude $a(t)$ of the shear wave can easily be found at each step of a simulation of this system, and the corresponding viscosity found through the relation

$$\nu = \frac{-1}{k^2}\frac{d}{dt}\ln a. \tag{A.7}$$

This can be compared with the theoretical value of the viscosity derived from the relaxation time τ.

Hele-Shaw flow

If a body force G is applied to fluid in a Hele-Shaw cell of width h, the fluid will equilibrate at a constant velocity v in the direction of the body force, where

$$v = \frac{Gh^2}{8\nu}. \tag{A.8}$$

If no-slip walls are added to form a Hele-Shaw channel of width L, then the velocity profile takes the form

$$v(x) = \frac{Gh^2}{8\nu}\left[1 - \frac{\cosh\left(\frac{2\sqrt{2}x}{h}\right)}{\cosh\left(\frac{\sqrt{2}L}{h}\right)}\right] \tag{A.9}$$

It is extremely straightforward in both of these cases to verify the equilibrium velocity peak.

A.9.3 Regression tests

Regression tests check that the program produces the same behaviour as it did previously. For example, if a performance optimization is made to the code without altering the simulated physics, it should generate identical results to an unoptimized version.

Regression tests were implemented on both LB2D and LB3D by running a simulation for a small number of timesteps, and then checkpointing its state to a file. On LB3D, the simulated system is a phase-separating ternary mixture; on LB2D a

phase-separating binary mixture. The simulation is then restarted from the saved state on a known-good implementation of the code, run for a few hundred timesteps, and the simulation state at the endpoint saved to disk.

Regression testing is then a simple matter of restarting a simulation from the saved checkpoint, re-running, checkpointing to disk, and performing a binary comparison of the saved endpoint state with the original saved endpoint: any differences indicate deviations from the original behaviour due to programming error.

This testing system was used not only to make sure that optimized versions of the code produced the same results, but also that the results did not change with number of processors used in parallel calculations. Comparing the results from a simulation restarted from a single master checkpoint file avoided possible variations due to randomized initial conditions.

Because the data (both scientific output and checkpoint data) is saved in the portable XDR format, comparisons of precisely the same simulated system may be made across different computing platforms. It was verified that the single-precision fluid density fields at the endpoint were identical, regardless of which architecture in table A.2 was used.

A.10 Parallelisation

A.10.1 Taskfarming

Taskfarming is the practice of running many concurrent independent simulations simultaneously. Because the simulations are independent, no communication is required between CPUs, so it is a particularly simple form of parallel processing to implement. However, each simulation must be small enough to fit on one processor, so taskfarming is only suited to running large numbers of small simulations, for example when performing searches of parameter space.

An N-way taskfarm calculation is typically performed by generating N different input files, one for each simulation to be performed. Then N different simulation processes are launched, each reading the appropriate input file and performing the corresponding simulation. Both LB2D and LB3D are able to run taskfarms using the standard Message Passing Interface (MPI)[244] library for parallel processing: in this case, N copies of the same simulation executable are started from the `mpirun` launcher. Each process calls `MPI_Comm_rank()` when it starts, and receives a number (the MPI "rank") between 0 and $N-1$ to indicate which input file it should load. This procedure allows taskfarms on any platform with the MPI libraries installed.

LB2D also supports taskfarming through the use of the UNIX `fork()` system call. In this case, a single LB2D is launched from the command line, which calls `fork()` $N-1$ times to launch the rest of the processes. This method has the advantage that it will work on any UNIX platform, regardless of whether or not MPI is installed.

A.10.2 Domain decomposition

A computer with a 32-bit address space can access up to four gigabytes of memory. A single LB3D lattice site without any porous medium data takes up a minimum of 480 bytes, so the absolute maximum size of simulation which can be performed on a 32-bit machine is around 200^3, although the practical maximum can be significantly

less, since memory is also required for the operating system, auxiliary programs, executable, and temporary buffers. At around 10^4 lattice updates per second, the simulation would take around three months to reach 10^4 timesteps: a timescale which is not completely impractical, but tediously slow for a single simulation, and not amenable to approaches such as computational steering.

Towards the end of the work described in this thesis, commodity 64-bit computers capable of addressing more than 4GB without any special measures had become available, but a large simulation run on such a machine would remain impractically slow.

Both the memory and speed limitations can be addressed by splitting a simulation across many computers, each working on its own subset of the problem and communicating with other machines as necessary. LB3D uses a block domain decomposition strategy (Figure A.5): the CPUs are arranged into a 3D Cartesian grid, and each allocated the corresponding cuboidal subset of lattice sites. Unlike, for example, Molecular Dynamics simulations, very little data needs to be shared between processors: the only communication required during a simulation timestep is the exchange of nearest-neighbour data between adjacent CPUs.

Extensive benchmarking of the code has been performed[31, 253, 254], showing that LB3D scales well up to at least 1024 CPUs. Suppose the smallest number of CPUs a problem can run on is n; if a particular calculation takes time T_n to run on n CPUs, and time T_N on N CPUs, then the "speedup factor" for the N CPU calculation is defined as

$$S \doteq \frac{nT_n}{T_N}. \tag{A.10}$$

Often, $n = 1$, so the speedup is the factor by which the calculation speeds up compared to serial performance. If $S = 1$, the calculation is said to scale linearly. LB3D was benchmarked by Kevin Roy of CSAR[253] and Elena Breitmoser of EPCC[254] on two different platforms, and found to show closely linear scaling up to 64 CPUs on the SGI Origin2000 platform, and good scaling up to 1024 CPUs on an IBM SP4, as shown in figures A.6 and A.7. The halo exchange routine was found to take a roughly constant 5% of the total elapsed time. For a 512^3 simulation on an IBM SP4, the elapsed time was reduced by a factor of 1.8 when going from 512 CPUs to 1024 CPUs — sufficiently good that LB3D was one of the first codes to be placed in the "Gold" scalability class on the HPCx supercomputer.

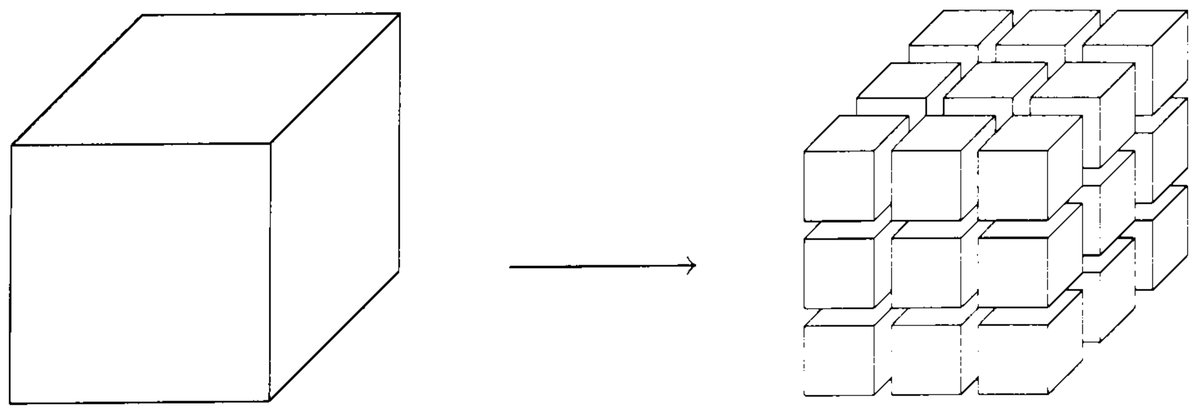


Figure A.5: Domain decomposition of an LB3D simulation: 27 CPUs are split into a $3 \times 3 \times 3$ grid, each CPU receiving an identically-sized block of the lattice.

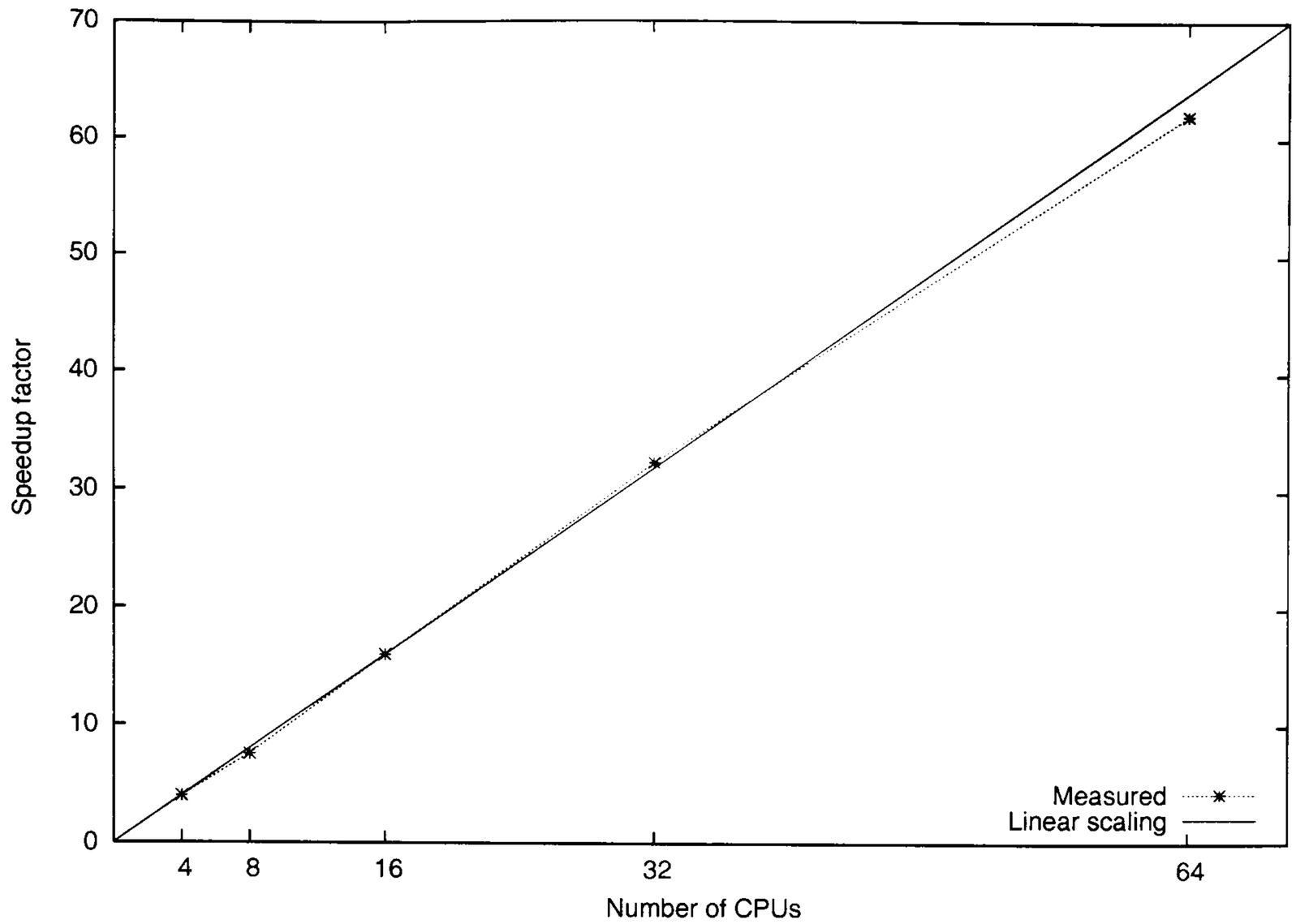


Figure A.6: LB3D speedup factor for 20 timesteps of a 256^3 system running on an SGI Origin 2000, writing data every 10 timesteps. Data taken from [253].

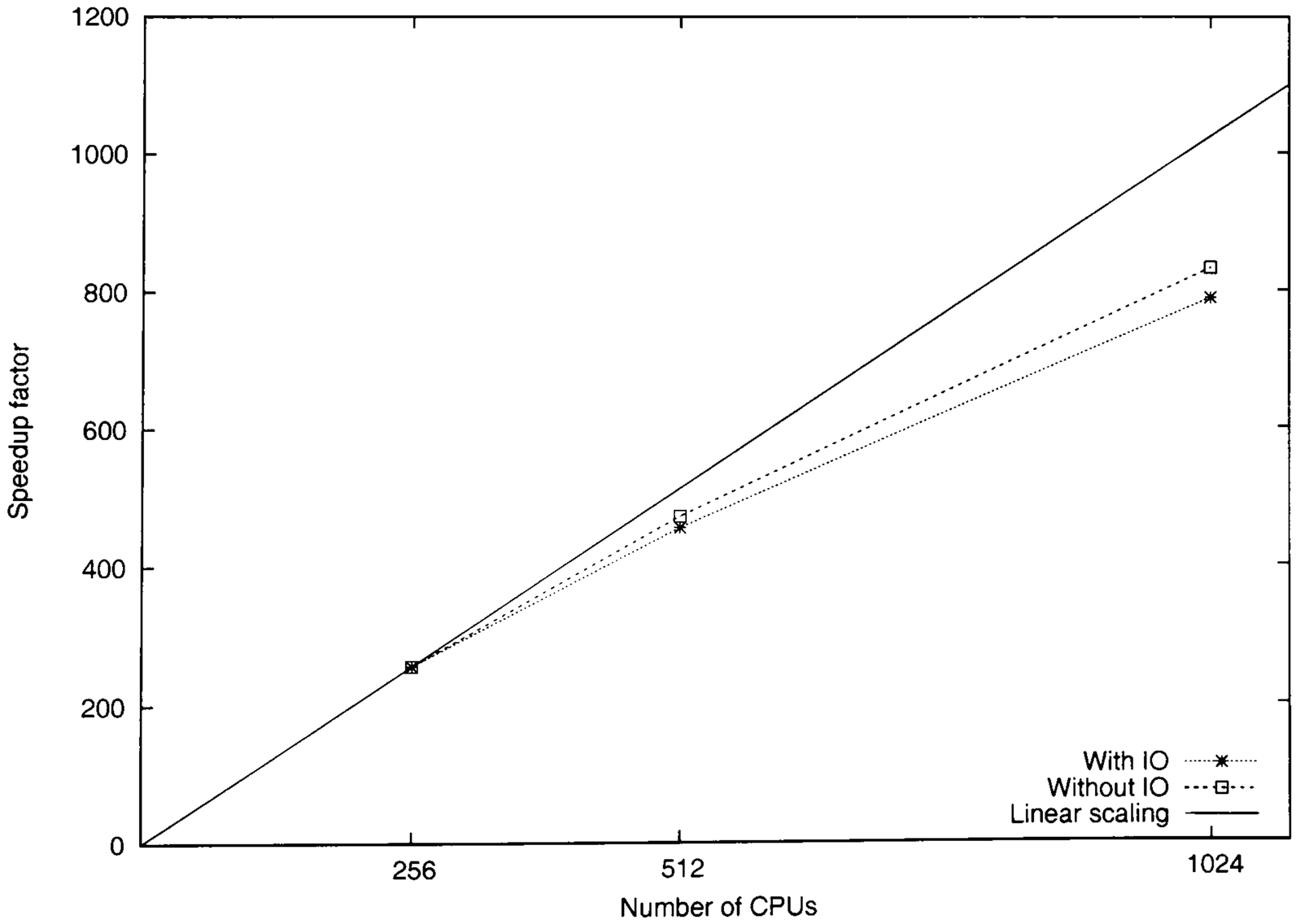


Figure A.7: LB3D speedup factor for 20 timesteps of a 512^3 system running on an IBM SP4, with and without data output every 10 timesteps. Data taken from [254].

A.10.3 Parallel IO

When running a single calculation distributed across many processors, there are several strategies for writing output from the simulation back to disk. Typically, one might wish to observe the values of a certain set of fields (such as velocity or composition) at regular intervals. The desired end result of a simulation is a set of files, one for each field, for each point in time to be examined. However, in a parallel calculation, the data to be written is distributed across the processors, and must be combined into a single output file. The appropriate manner in which to do this can depend strongly on the architecture on which the simulation is being performed.

Distributed IO with postprocessing

A parallel task using the MPI library is generally offered no guarantees about the nature of the filesystem available to each processor. On one extreme, large supercomputers can give all CPUs fast access to a common shared filesystem; on the other extreme, clusters built from commodity components may have their disk storage distributed across nodes with the consequence that access to the disk storage of a remote node, if possible at all, must go through a relatively slow ethernet link. Many systems are somewhere between these two extremes.

The simplest approach to IO is also the most portable, since it requires no guaranteed global filesystem. In this approach, each CPU writes its own subset of the relevant data to disk when required, and then continues with the simulation. After the simulation has terminated, a postprocessing program is then run, which collects the data from each CPU, collecting it together to produce a single file for each field for each timestep, as shown schematically in figure A.8.

This procedure has the advantages that it makes no assumptions about the availability of a common filesystem, and also that it is possible for conversion to the portable XDR format to occur in the postprocessor stage. This means that a Fortran XDR library is not required, although it does require the postprocessor to understand the Fortran binary output format. However, this approach means that the results from a simulation only become available once the entire simulation has terminated: this makes it difficult to monitor the simulation while it is running, and makes computational steering all but impossible. This was the first IO mode to be implemented in LB3D (since it corresponds to how IO was done in ME3D), and remains as a fallback for when IO is not possible through other modes.

Internal postprocessing

The next simplest approach, which also has no requirement for a shared filesystem, is to gather all of the output data on one CPU before writing it to disk (Figure A.9). This removes even the requirement that all CPUs must have access to disk storage: only the CPU which gathers the data needs this, although determining which CPU this is can be a complicated and very system-dependent operation.

However, this operation requires that the CPU with disk access also have enough free memory to hold the entire output file in memory at the same time as its own simulation data, which may not be the case for large systems. In addition, it requires the calculation to pause while a very large amount of data is communicated to a single CPU, which can then become a severe performance bottleneck[253].

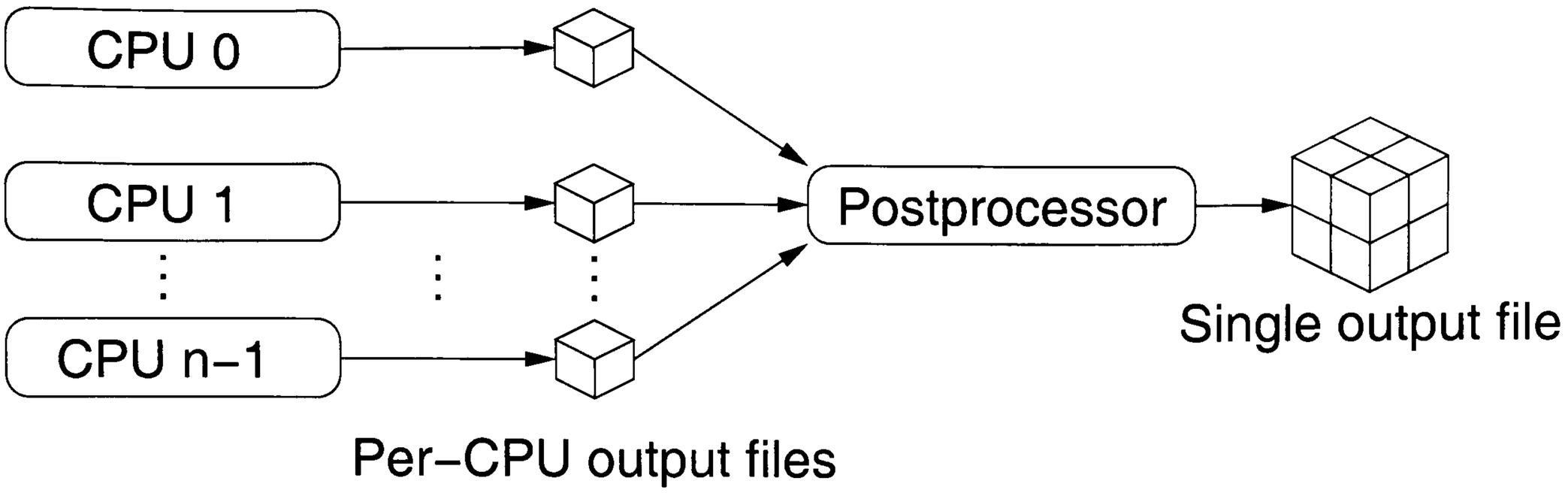


Figure A.8: The simplest approach to parallel IO is for each CPU to write its own subset of the relevant data to disk; this is later combined together by a separate postprocessor.

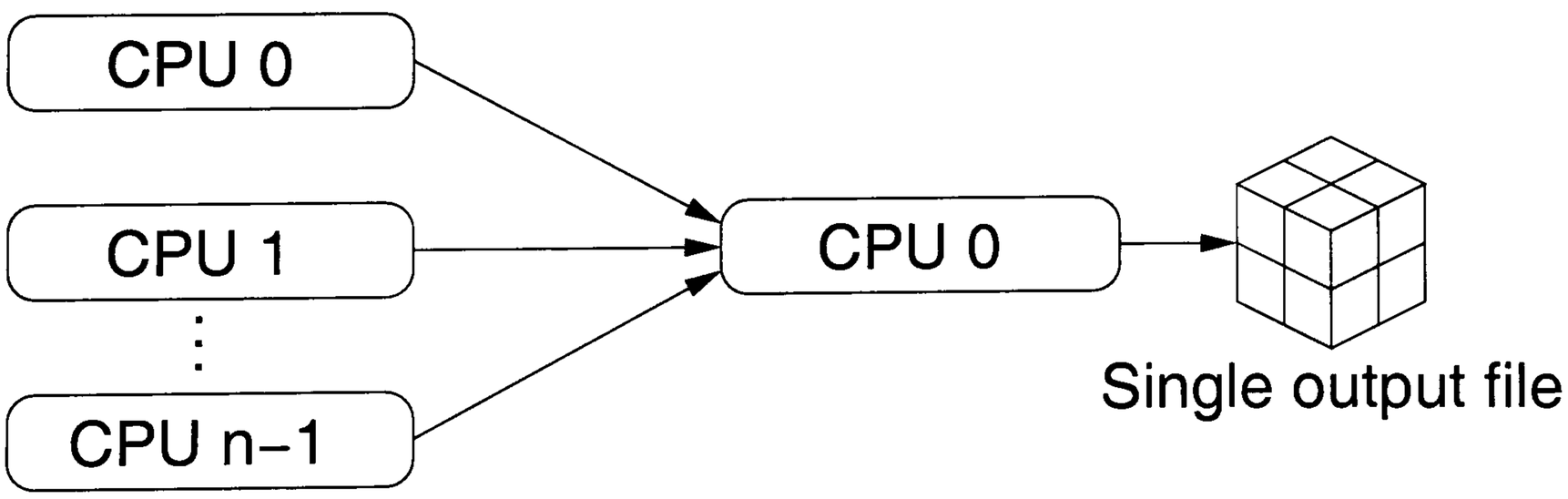


Figure A.9: The next simplest approach to parallel IO is for all data to be collected on one CPU before being written to disk.

Fully parallel IO

The most sophisticated output mode supported by LB3D is also the most demanding in terms of hardware and software requirements. In this mode, when IO is performed, a single file is written; each CPU writes into the appropriate section of the file. which is then closed when all CPUs have written their output (Figure A.10). This eliminates the need for any postprocessing, and can easily cope with the case where the entire output dataset will not fit into the memory of one CPU; however, it also requires access to a shared filesystem for all CPUs.

In LB3D, this IO mode was added by collaborators at EPCC, who used the HDF5 software library to do so. The HDF5 library provides access to a portable file format similar to (but more complicated than) XDR, and uses the MPI-IO layer of version 2 of the MPI standard to actually perform the parallel IO. It is not uncommon for supercomputer vendors to have their own specially optimized implementation of MPI. Whereas a program which used its own parallel communication and IO systems would have to be ported to and optimized for each new architecture on which it is to be run, programs using MPI and MPI-IO can be ported with a minimum of changes (often no changes at all), and can take advantage of the optimizations in the locally installed version of MPI without having to be optimized themselves.

A.11 Auxiliary tools: `vol3d`

Much of the analysis of data from LB3D was performed using a library called `vol3d` , developed during the TeraGyroid project. Initially, `vol3d` was written as a tool for interchange between data formats: LB3D can write output in HDF5 and XDR formats, but the various tools for data analysis and visualization may require data to be in raw floating-point format, or other formats such as VTK or NRRD. Many of these formats are really just trivial rearrangements of the metadata, but a tool for automatic conversion between them is necessary for dealing with the volumes of data produced by LB3D.

`vol3d` quickly grew from an interchange tool to a general-purpose analysis utility. Its main design philosophy is that all the scientific data from LB3D — density fields, composition fields, velocity fields, Fourier transforms of such fields, etc. — can be treated as 3D Cartesian arrays of some datatype. In the case of density or composition fields, this datatype is a simple scalar; in the case of velocity fields, a 3-vector.

Operations on these datasets can be divided into those which require no information about what is actually inside each element of the 3D arrays (such as memory allocation and deallocation, copying, transposition, and generation of subsets of the data), those which consist of an operation on a single element, repeated pointwise across the whole lattice, and other operations which need more information. Operations in the first class can be implemented once and then used for all possible datatypes. To support operations in the second class, iterator methods were written which would take a function which operates on a single element, and then apply it across the array. To implement the operation of negating all points on the (floating-point valued) array, one needs only to write the trivial function which negates a single floating-point number; the iterator then takes care of applying it in the appropriate way. This made it very easy to rapidly try out different image processing algorithms on the TeraGyroid data.

`vol3d` was written as a C library; several tools were built on top of this to expose its functionality to the end user. This approach of separating the main number-crunching code from the user interface, as well as improving maintainability, has the advantage that any other applications which need to load or save 3D data in many formats can simply link to the `vol3d` library rather than reimplementing or duplicating any of the `vol3d` code. The program which was used to generate most of the statistical data (such as the Euler number time series) from the TeraGyroid datasets was a mere 60 lines long as a result, since it consisted mostly of straightforward library calls. Several other tools, for data manipulation and interactive visualization, were also built on top of `vol3d` .

`vol3d` was written with the same portability strategy as LB2D: it is entirely written in ANSI C, uses GNU Autoconf for compile-time configuration, and supports automatic detection and support of auxiliary libraries such as FFTW[255], `libpng`, and HDF5; it therefore enjoys the same level of portability.

A.12 Computational Steering

A.12.1 Traditional simulation methodology and its drawbacks

Traditionally, large, compute-intensive simulations are run non-interactively. A text file describing the initial conditions and parameters for the course of a simulation is prepared, and then the simulation is submitted to a batch queue, to wait until there are enough resources available to run the simulation. The simulation runs entirely according to the prepared input file, and outputs the results to disk for the user to examine later.

This mode of working is sufficient for many simple investigations of mesoscale fluid behaviour; however, it has several drawbacks. Firstly, consider the situation where one wishes to examine the dynamics of the separation of two immiscible fluids: this is a subject which has been of considerable interest in the modelling community in recent years [181, 96]. Typically, a guess is made as to how long the simulation must run before producing a phase separation, and then the code is run for a fixed number of timesteps. If a phase transition does not occur within this number of timesteps, then the job must be resubmitted to the batch queue, and restarted. However, if a phase transition occurs in the early stages of the simulation, then the rest of the compute time will be spent simulating an equilibrium system of very little interest; worse, if the initial parameters of the system turn out not to produce a phase separation, then all of the CPU time invested in the simulation will have gone to waste.

Secondly, the input file often takes the simple form of a list of parameters and their values, but this may not be sufficiently expressive to describe the boundary conditions one may wish to apply, or the conditions under which they are to be applied. For example, to simulate the flow of a fluid mixture through a porous medium, it can be necessary to equilibrate the flow of a single component through the medium first, before introducing the fluid mixture, in order to prevent the behaviour of the mixture from being affected by transients present as the flow field develops.

A.12.2 Non-interactive steering: scripted simulations

For every new and complicated boundary condition one wishes to impose, it is in principle possible to write a corresponding new subroutine in the simulation code, add an option in the input file to switch this boundary condition on or off, and recompile the simulation code. However, in practice this leads to redundancy and overcomplication, or "bloat", in the simulation code, and also to excessively complicated or verbose input files. Bloated code will, in the long term, become difficult to maintain or change, and more complicated input file syntax makes it harder for new users to learn how to use the code.

An alternative strategy is to abandon the concept of an input file altogether, and instead to control the simulation from a script written in a high-level language. This has several advantages.

Firstly, provided that enough access to the simulation data structures is provided to the scripting layer, new boundary conditions may be formulated, tested, and run with ease, without requiring the code to be recompiled. The core of the number-crunching code stays small and maintainable as a result.

Secondly, a high-level language provides conditional and loop structures, so simulations may be given much more detailed instructions than simply to run for a fixed number of timesteps: for example, a simulation of fluid phase separation could be instructed to run until the fluid components have separated to a certain degree, and to then stop.

Thirdly, writing the core of the simulation code in a language like C or Fortran but controlling its behaviour through a higher-level language allows the programmer to easily interface the simulation code with other components (such as image generation libraries) via the high-level language, which avoids the necessity of dealing with tedious low-level details of interfacing to many third-party libraries. This strategy also avoids incurring the performance penalty that would result from writing the entire simulation code in a higher-level language.

The approach of making a piece of code such as the simulation solver available as a self-contained reusable object to some higher-level "glue" layer is often termed "componentization". In this case, the glue layer is the high-level language; in the more general case, it could be a Grid fabric layer such as Web Services, allowing interoperation across the network of components running on different machines.

The high-level language chosen to script LB2D was Perl, a powerful language popular, amongst other things, for its ability to interface with, or 'glue', external components, and also for the wide variety of freely-available Perl code which can be easily accessed from scripts written in the language [256]. Constructing a functional Perl interface to the simulation code required little more than writing a formal description of the C subroutines comprising the simulation code [257]; interfacing code to other popular scripting languages such as TCL, Python, or Ruby is typically just as easy.

A.12.3 Parameter space exploration through high-level scripting

For an algorithm which runs the risk of encountering numerical instabilities, it is desirable to know the regions of parameter space in which one can operate without expecting to encounter such problems: for example, when studying the behaviour

of an interface between two fluid components, it is useful to know how high the surface tension can be set before numerical instabilities are introduced, resulting in a simulation crash.

A crude approach to map out this region is to guess the size and location of a region of parameter space which will contain a region of stability, and blindly launch many simulations over this space.

A slightly more versatile approach made possible by scripting is to start with a known-stable point in parameter space, and an initial direction in parameter space. A simulation is started at the known-stable point, and if it completes successfully, another one is started at an adjacent point in parameter space, until eventually a numerically unstable régime is found; successive simulations can then be launched to home in on the location of the stable/unstable boundary.

In the Shan-Chen LB model, the interaction force between components gives rise to a surface tension at the interface between regions of different components; the strength of this interaction, and therefore the magnitude of the surface tension, is controlled by the coupling constant g_{cc}. A simple parameter-space investigation is to take an interface between two immiscible fluids, and run simulations with increasing g_{cc}, and therefore increasing surface tension, until numerical instability sets in. The script controlling this process ensures that each simulation runs for long enough that the system reaches equilibrium. If a simulation succeeds, then the surface tension is raised; if not, then it is lowered, and the boundary is located using an interval bisection algorithm. This procedure was used to quickly establish a stable parameter space before commencing other simulations using LB2D.

A.12.4 Scripted boundary conditions

The use of scripting allows extremely versatile and dynamic specification of boundary conditions in a simulation. Consider a stream of droplets flowing through a channel until they meet an obstacle: the droplets accumulate on the obstacle and coalesce to form a larger droplet, until the resulting droplet becomes too large and breaks free to travel further down the channel. Conventionally, one might investigate this situation by first running a simulation of single-component flow through the channel with the obstacle, until the velocity field equilibrates, and then starting a new simulation from this one in which droplets are added to the channel, either through manual intervention, or by writing an additional piece of code which periodically generates a new droplet near the entrance to the channel.

However, it is also possible to perform the simulation in one single run, using a script. This script equilibrates the single-component velocity field, and then automatically introduces a droplet to the channel entrance. It then waits, monitoring the simulation state, until the droplet has moved sufficiently far down the channel that a new one may be introduced without colliding with it. The results of such a simulation are shown in Figure A.11: the automatically-generated droplet stream is induced by the obstacle to coalesce into a single large droplet, which then breaks off shortly before timestep 7850. The advantage of this approach is that it does not require human intervention to restart the simulation after the velocity field has equilibrated, nor does it require a fixed droplet generation rate to be set: if such a rate were set slightly too high, then droplets may collide with one another before reaching the obstacle, a situation avoided by the use of a dynamically-specified boundary condition.

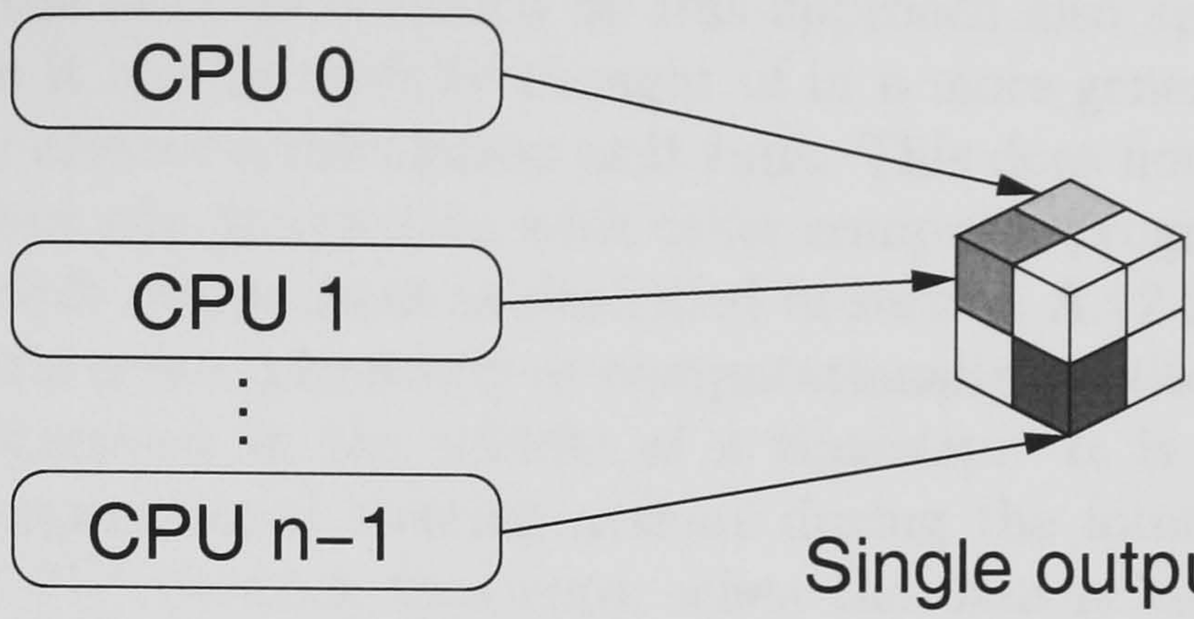


Figure A.10: A parallel IO system permits each CPU to write its chunk of the final dataset to the appropriate part of the resulting output file.

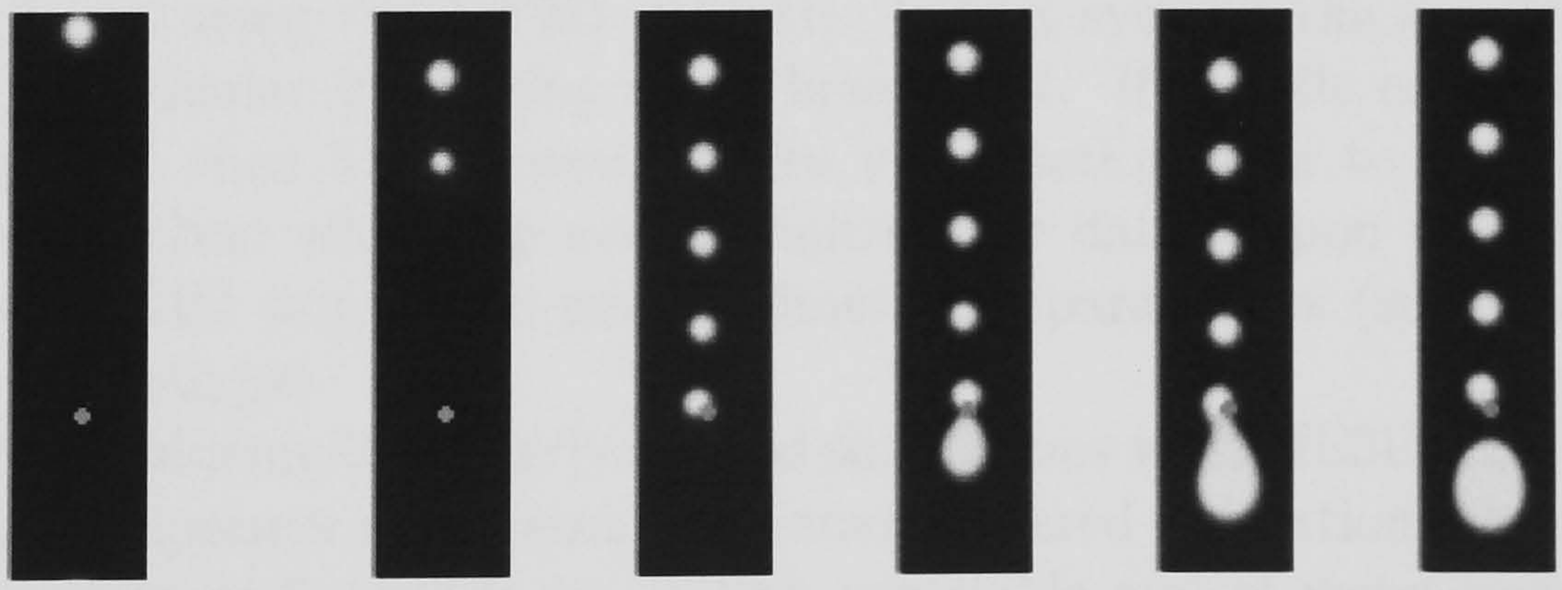


Figure A.11: Simulation of droplet coalescence on an obstacle during channel flow using scripted boundary conditions

A.12.5 Interactive Computational steering

Computational steering is defined by Gu *et al* [258] as "the capacity to control the execution of long-running, resource-intensive programs". This definition emphasises the nature of computational steering as an alternative to the traditional offline batch-queue mode of operation for high-performance computing applications; however, many of the benefits obtained by this approach also apply to less demanding applications, so it can be usefully thought of in a more general sense as the ability to interactively control a calculation as it runs. This does not just mean interaction with humans, but also interaction with other computer programs, such as scripting languages or other components as described in section A.12.2.

It makes little sense, physically or computationally, to alter the state of a lattice Boltzmann calculation in the middle of a timestep. It is natural, therefore, to perform all computational steering actions during the interval (sometimes called a "breakpoint"[71]) between timesteps, when the data in the simulation lattice is self-consistent. In LB3D, operations which can take place at a breakpoint include the pausing, resumption, or checkpointing of a simulation, adjustments of physical parameters such as coupling constants, or adjustments of parameters such as data output rate.

The most obvious way to implement computational steering in a simulation would be for the simulation code to check for user input at the keyboard every timestep, and take action accordingly if any is found. However, in a supercomputing environment where most calculations run from a batch queue, this method would not work, since the simulation program would not have access to the user's keyboard or login session. Moreover, control of terminal-based IO, especially from Fortran, can be subtle and complicated to program.

Given the aforementioned lack of standardized Fortran interfaces to UNIX interprocess communication (IPC) systems such as pipes, sockets, or shared memory, the easiest way for a simulation code to communicate with other programs or users is through the filesystem.

The use of this style of computational steering in mesoscale fluid modelling was pioneered by Love, using the ME3D code[31]. In this system, the simulation code would check a particular file during every breakpoint. If the file contained an instruction to pause, then ME3D would write visualization data to disk, and then suspend the simulation while the user visualized the data. Upon receiving a signal to resume, ME3D would also permit simulation parameters (such as coupling constants) to be changed.

It became clear during these early steered simulations with ME3D that visualization is a crucial component of any computationally steered simulation: the structural behaviour of mesoscale fluids can be at best unreliable and at worst impossible to automatically quantify, whereas the human eye can pick out what is happening very quickly, presented with, for example, a rendering of a fluid interface. However, the file-based steering system in ME3D required that the simulation and visualization be performed on the same machine, or on machines with a shared filesystem. This can be a limitation for large-scale simulations, since the machine performing the simulation may not have enough spare capacity to also perform the visualization, or it may not be equipped with accelerated graphics hardware, forcing the rendering to be performed much more slowly in software.

As part of the RealityGrid project[259], a computational steering facility was added to LB3D, in the guise of the RealityGrid steering library[260]. The library

was developed as an independent piece of code which could be linked into any simulation program to make it steerable. This gave LB3D the capability to perform not only file-based steering, but also networked steering from a remote machine. Further details are presented in section B.3.2.

A.13 Conclusions and future work

The design of LB2D, with the core number-crunching code written in C, but higher-level functionality written in Perl, made analysis of results, automatically steered simulations, and interfacing to other libraries and pieces of software significantly easier, without incurring a large performance penalty. The encapsulation of the lattice Boltzmann code into an opaque object made maintenance of the higher-level functions much easier, since they were insensitive to changes to the internal structure. Indeed, it ought to be possible to go as far as implementing a sparse-lattice LB code in this framework without having to change the external API.

The functions which were added to the LB codes (such as scripting, steering, image generation, etc) beyond the simple number-crunching level all required quite sophisticated IO and communication operations, which were at times awkward to perform portably using Fortran.

The gyroid data analysis tools benefited from the design philosophy that all lattice Boltzmann data, and a very large part of the data derived from lattice Boltzmann simulations, consists of Cartesian arrays of (often opaque) objects; further use of this abstraction could similarly benefit a new lattice Boltzmann code.

It took significantly longer to analyse the gyroid simulation data than it did to generate it: this underlines the idea that lattice Boltzmann codes (and simulation codes in general) should be regarded as single parts of a much larger computational system including experiments, simulation, and analysis. It took an extra sixteen lines of code in the `vol3d` library (rather than the creation of an entirely new conversion program) to permit the gyroid simulation results to be viewed in a third-party volume rendering package[226] which permitted visualization of the defect regions: in this sense, what a simulation code can be plugged into has as much impact on its scientific utility as its performance.

The benchmarks of LB3D show that lattice Boltzmann is an extremely scalable algorithm when implemented appropriately on distributed-memory parallel processing architectures, although the appropriate choice of IO strategy can vary from machine to machine.

Parallelization of lattice Boltzmann follows a Single Instruction, Multiple Data (SIMD) approach: each CPU applies exactly the same algorithm (advect, then collide) to different chunks of the lattice: this algorithm is sufficiently simple and parallelizable that one might even consider implementing it in hardware. Margolus and Toffoli[261, 262] pioneered hardware architectures to run cellular-automaton models of physics, but such designs have not to date become widespread, possibly because any performance gain obtained by a specialized hardware implementation is rapidly eclipsed by the exponential growth in power of general-purpose microprocessors. Moreover, the LB algorithm requires the use of floating-point arithmetic, which is costly to implement. However, inexpensive but powerful parallel SIMD processors have recently become widely available in the form of the Graphics Processing Units (GPUs) on graphics cards. Although these devices are designed for the fast processing of various geometrical operations when rendering 3D images, it has

long been known[263] that it is possible to express cellular automaton algorithms using these operations, and in doing so, benefit from the hardware speedup. It has recently been shown[264, 265] that it is possible to implement lattice Boltzmann on a GPU, potentially resulting in performance gains which would allow access to hitherto inaccessible physical régimes.

Appendix B

The TeraGyroid Project

The results presented and analysed in chapter 4 were generated as part of the TeraGyroid project[196]. The following chapter presents an overview of the project, and some of the technical issues which arose over its course.

The aim of the TeraGyroid project was to link two computational Grids — the UK e-Science Grid and the US TeraGrid — in order to perform large-scale calculations which would not be possible otherwise. The gyroid domain dynamics calculations were identified as an ideal candidate for the project, due to their ambitious scale, and due to the significant amount of work which had already taken place to adapt the LB3D code to work in Grid environments. Essentially, the project had to solve the problem of running a set of very large simulations over resources which were distributed around the globe, and subject to varying availability.

B.1 Problems posed by the project

The principal problem to be dealt with was the very large system sizes which had to be simulated in order to reach the physical régime of interest. This led to very demanding requirements not only for raw computing power, but also for visualization, storage, and management facilities.

B.1.1 Simulation requirements

Memory and CPU

The gyroid phase had been observed in simulations of up to 64^3, at timescales up to order 10^4 timesteps. It was therefore planned to run simulations on system sizes of 128^3, 256^3, and 512^3 to determine whether finite size effects are still present at larger system sizes.

The ternary amphiphilic version of the LB3D code runs at a minimum of 10^4 lattice site updates per second per CPU on most modern computer architectures, and has been observed to have roughly linear processing up to order 10^3 CPUs [31, 71]. For example, a 128^3 lattice comprises 2.1×10^6 sites; this means that running it for 1000 timesteps would require 2.1×10^9 lattice site updates. which would take around 2.1×10^5 CPU seconds. Across 64 CPUs, this would take 3.3×10^3 wall-clock seconds, or about an hour of real time.

LB3D required approximately one kilobyte of memory per lattice site: hence, a 128^3 system would require about 2.2GB of total memory to run; a single system

System size	Number of lattice sites	Approximate system state size	Scientific visualizatic dataset size
64^3	262144	300 MB	1 MB
128^3	2.1×10^6	2.1 GB	8.5 MB
192^3	7.08×10^6	7.07 GB	28.3 MB
256^3	1.68×10^7	16.8 GB	67.1 MB
512^3	1.34×10^8	134 GB	537 MB
1024^3	1.07×10^9	1.07 TB	4.3 GB

Table B.1: Memory requirements for different LB3D system sizes

checkpoint would then require about 2.2GB of disk space.

A floating-point scalar field dataset for visualization takes four bytes per lattice site, or around 8MB per dataset on 128^3. Each timestep whose state was to be visualized required emission of at least one scalar dataset; each point at which a simulation was steered (ie parameters changed in real-time) required a complete system state to be saved to disk. The memory requirement to hold the simulation state, and the amount of disk space for a single scalar dataset, are listed in table B.1: the problems sizes to be tackled were all sufficiently large that distributed-memory multiprocessing was the only viable option.

Visualization

At the outset of the project, a typical workstation was able to render datasets up to 64^3 using either isosurfacing or volume rendering: for example, a machine with a GeForce3 Ti200 graphics processor and AMD Athlon XP1700 CPU can render a 64^3 gyroid isosurface containing 296606 triangles at around 5 frames per second. Larger datasets would take several minutes of pre-processing, and then require several seconds to render each frame, resulting in a jerky display which made user interaction and location of regions of interest rather difficult. Furthermore, datasets of 256^3 and larger would sometimes simply not fit into the available graphics memory on a single workstation. Parallel rendering techniques were therefore required.

TeraGyroid was a collaborative project involving many institutions in the UK and the USA. In particular, the people launching and steering the simulations and visualization were located in London and Ipswich in the UK, and in Massachusetts and Arizona in the USA; all of these people needed to be able to see what was happening during the simulation runs, and to discuss the running simulations with one another. In addition, it was desirable to be able to demonstrate the steered simulations to an audience at the Supercomputing 2003 conference, and to a worldwide audience at the SCGlobal teleconference.

Data storage

The simulation checkpoints alone could take up to two terabytes of space; the data output from the simulations was expected to take a similar amount of space. This required both a large amount of free disk space on the simulation machines, and a large amount of long-term storage to hold the postprocessed results.

Computational steering

Because of the nature of the calculations and the resource constraints imposed upon them, it was thought that the TeraGyroid project would be an ideal candidate for computational steering. Due to a poor theoretical understanding of the problem, and particularly of its dynamical nature, it was not known beforehand how long the simulations would have to run before reaching a gyroid domain régime, nor how long that régime would last, nor even which regions of the simulation parameter space would and woujld not permit gyroids to form. It was hoped that computational steering would allow simulations to be monitored as they ran: those which crashed or entered uninteresting régimes could be terminated early, while those which showed particularly interesting results could be adjusted to give higher resolution data output.

The resources allocated to the project were quite "bursty", consisting largely of timeslots (ranging from six hours to three days) during which all (or a large part) of certain machines would become completely available to the project. In a conventionally-operated batch queueing system, one does not have to worry about trying to fill up a whole machine with simultaneous tasks: a sufficiently well-designed queueing system will simply allow other peoples' tasks to take up unused resources. This was not the case for TeraGyroid. The combination of the somewhat idiosyncratic resource allocation and the steered jobs meant that at any time, calculations could be running out of time on one machine, while processors on another machine (potentially on another continent) were being freed by a terminating job. This led to the desire for migratable simulations, which could move (or at least, be moved) from machine to machine in order to make best use of the available processors.

B.2 Computational Grids

The concept of "computational grid" emerged from the field of distributed computing. Computational grids (or simply, "Grids") were initially[266] defined by analogy with the electrical power grid, the idea being that with sufficient improvement in bandwidth, connectivity, and the software to manage them, computer resources could become obtainable on-demand in much the same way that one can obtain electrical power resources simply by plugging an appliance into the wall socket that connects it to the electrical grid.

In their influential tome, Foster and Kesselman[266] define a computational grid as " a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities". This reflects an early emphasis on High-Performance Computing (HPC) aspects of Grid computing, in the hope that developments in the field would both widen access to HPC resources, and also allow HPC resources at multiple sites to be harnessed together to tackle very large problems, such as the 10 Petabytes of data which, it is estimated, will be generated by the Large Hadron Collider project.

From the perspective of the computational scientist, the motivation for Grid technology was that it should simplify the process of running a simulation. Traditionally, in order to run a calculation which needed serious amounts of computing power, one needed first to acquire time on a suitable computer, then log into that computer, build the simulation code there, launch a job, wait for it to run, then copy the results back to a local workstation or onto tape. Data visualization might ne-

cessitate travelling to the physical location of a specialized visualization computer. It was hoped that with the advent of Grid computing, a job could be submitted "to the Grid" from a user's workstation. A layer of software, called "middleware", would then take the job, and find a computer connected to the Grid which currently had enough capacity to run it; the job would then be compiled and run on that machine without the user even needing to know which machine it is or who owns it. The results from the job could then be stored on similarly transparently-accessible remote disk storage, or delivered back to the workstation. Just as one does not care which power station is generating the electricity used when plugging an appliance into the wall, or whether the power station is nuclear, coal-fired, or hydroelectric, the ideal for Grid computing was that a user should not have to care which machine a job runs on, or what architecture the machine is.

The turn of the millennium saw the rise of many distributed computing projects [267] taking many different approaches, but all using the term "Grid". Foster[268] attempted to clarify the situation by refining his definition of a Grid to "a system that coordinates resources that are not subject to centralized control, using standard, open, general-purpose protocols and interfaces, to deliver nontrivial qualities of service", reflecting a more recent shift in focus from HPC and a thin-client philosophy, towards a general concept of peer-to-peer distributed computing services which can be deployed together across administrative domains, and also a shift towards linking machines together with the Web Services protocol stack[269] rather than customized toolkits[270, 271].

"Web Services" is the name given to a set of protocols to allow different pieces of software to communicate over the Internet, through XML[272] message passing. Typically, this message passing takes the form of "remote procedure calls", allowing a program running on one machine to invoke a subroutine which actually exists inside a different program running on a different machine. However, the standard Web Services protocols do not specify whether a web service can hold internal information, or "state", or how that internal state is permitted to change between procedure calls. An attempt to address this issue within the wider context of Grid computing led to the concept of the Grid Service: roughly, any computational resource (stateful or otherwise) which exposes itself through a Web Services interface. Designing consistent stateful web services is a significantly more difficult task than designing stateless ones, not only because one must ensure that all state is kept consistent where necessary, but also because such services may only have a finite lifetime, necessitating careful construction and destruction semantics. The Open Grid Services Infrastructure (OGSI)[273] was an attempt to extend the basic Web Service protocols to include stateful, Grid-like interactions, but this was superceded by the WS-Resource Framework[274], amongst others.

The problems faced by the TeraGyroid project were seen as ideal candidates for solution through computational grid techniques, due to their scale, and to the distributed and rapidly changing nature of the available resources.

B.3 Available resources

B.3.1 Computation

The machines which were available for use are listed in table B.2. It was found that the machines varied the most with regards to how memory was treated: Green and

Machine name	Location	Architecture	No. CPUs
HPCx	Daresbury, UK	IBM Power4 Regatta	1024
Lemieux	PSC	HP/Compaq Alpha	3000
TeraGrid cluster	NCSA	Itanium2	512
TeraGrid cluster	SDSC	Itanium2	256
Green	CSAR	SGI Origin 3800	512
Newton	CSAR	SGI Altix 3700	256

Table B.2: Machines used for computation during the TeraGyroid project. PSC is Pittsburgh Supercomputing Center; CSAR is the UK National HPC Service in Manchester; NCSA is the National Center for Supercomputing Applications in Illinois, USA; SDSC is the San Diego Supercomputing Center in California, USA.

Newton, while using completely different CPUs, were both (non-uniform access time) shared-memory machines, with 512 and 384 gigabytes available in total, respectively, split evenly across CPUs. Lemieux was divided into nodes, each containing four CPUs and 4GB memory: a job could only be allocated an integer number of nodes. HPCx was similarly divided into 160 Logical Partitions or LPARs, each containing 8 CPUs and 8GB memory. Because these machines allocated an integer number of nodes to each job, rather than an integer number of CPUs, it turned out that the optimal configuration for certain simulation sizes was to leave a certain number of CPUs in each node idle, while using all of the node's memory.

B.3.2 The RealityGrid computational steering library

The RealityGrid steering library[260, 275, 71] is a piece of software designed and implemented by the Supercomputing, Visualization, and e-Science group at Manchester Computing, with the aim of making it as straightforward as possible to make existing simulation codes (such as LB3D) computationally steerable. It was designed so that the developers of a simulation code could link in the steering library, without having to know the intimate details of how the steering is actually performed. The library has been used with LB3D, and is also currently being used to investigate steered molecular dynamics simulations.

The steering library permits certain simulation parameters (such as lattice size or CPU time per timestep) to be monitored but not changed, and other parameters (such as coupling constants) to be changed while the simulation runs. If a simulation is launched with the steering library switched on, then at each breakpoint in the simulation, the steering library will check for steering commands. This means that such simulations can quite happily run unsteered, but that at any point, a steering client can attach to a running simulation to examine and possibly adjust it. Steering commands can be sent from the client to adjust parameters, to halt, pause, or resume the simulation, to force an output of visualization data, or to make the simulation checkpoint its state to disk. Communication between the client and the simulation is possible through files on a shared filesystem, or through exchange of XML documents through a network connection.

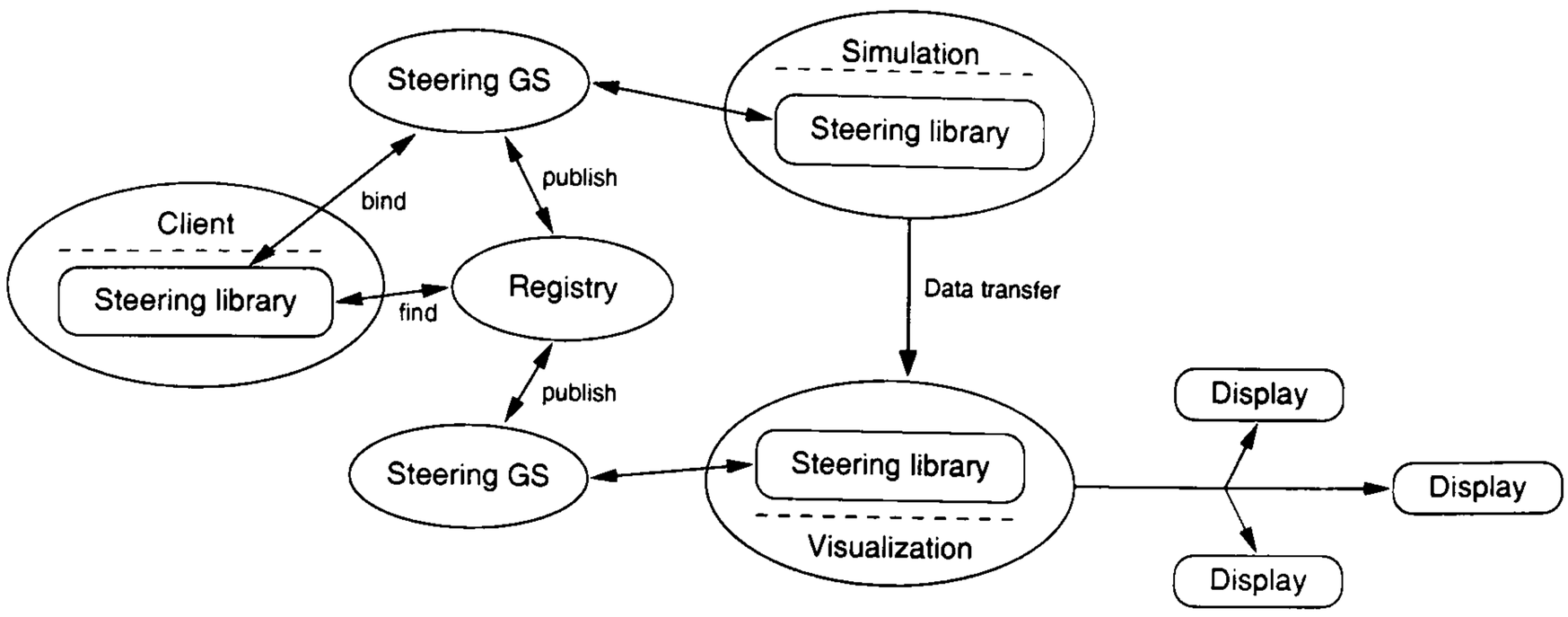


Figure B.1: The TeraGyroid Grid Service architecture

B.4 The TeraGyroid steered simulation architecture

The resources for TeraGyroid were tied together and coordinated through a variety of approaches, the most successful of which was the use of OGSI-oriented Grid services built using the `OGSI::Lite` toolkit. The overall structure of the system is shown in Figure B.1. When a steerable simulation is launched, the steering library creates a small grid service, the Steering Grid Service (SGS), to act as a "whiteboard", allowing messages to be passed asynchronously between the client and the simulation. The SGS in turn connects to another Grid Service, the Registry, which keeps track of and publishes the location of all Steering Grid Services. While the simulation is running, a client can conenct to the registry, find the SGS of the appropriate simulation, and then communicate with the simulation through the SGS.

Since the visualization process can be regarded as a calculation like any other, the code for visualizing the output from the simulations was also linked with the steering library, and can be controlled like any other steerable process. Data transfer between the simulation and visualization processes was accomplished using the `globus-io` library. Rendering was performed using a version of the VTK[224] toolkit, modified wot work with the RealityGrid steering utilities. The rendering was parallelized using the Chromium[276] toolkit for harnessing multiple graphics processing units (GPUs) in parallel, and the SGI Vizserver[277] software was used to allow interaction with the visualization. The rendered images from the visualization could be displayed on the workstation of the person performing the steering, or could be sent, via the FLXmitter multicast interface library[245], to the screen of anyone involved in a teleconference session. As well as allowing all participants in the project to watch and discuss the simulations as they ran, this meant that the simulations could be shown to a worldwide audience at the SCGlobal multicast conference.

B.4.1 Migratable simulations

Due to work by collaborators at EPCC, LB3D has the ability to perform "malleable" checkpoints: that is to say, a simulation running on one machine can be stopped and checkpointed, and then restarted on a different number of CPUs on a completely different machine. This permits simulations to be migrated between different HPC facilities, simply by connecting a steering client to a running simulation, issuing

stop and checkpoint commands, copying the checkpoint files over using the GridFTP protocol, and then launching a restarted simulation on the new machine. The details of the networking issues when performing this procedure are examined by Pickles *et al* [196].

B.4.2 Organizational details

When steering a simulation, it became common practice to take a checkpoint immediately before performing the steering action. This meant that if the steering caused the simulation to crash, it could be "rewound" back to its last known valid state, and the calculation continued from there. While this approach means that simulation crashes are much less of a hindrance while performing steered parameter searches, it has the drawback that it tends to produce a large number of checkpoint files, all of simulations at slightly different states which evolved through slightly different paths, and keeping track of these can involve a large amount of administrative work.

This administrative overhead was dealt with using the "Checkpoint Tree" Grid Service, developed by Manchester Computing. This is another Grid Service which not only keeps a record of each generated checkpoint, but which simulation it was generated from, and which steering actions led to its generation. Effectively, this means that any simulaiton can be "replayed" from a checkpoint earlier in its life, and removes the administrative burden from the scientist performing the steering.

The project involved a very large number of people working on two continents and across several timezones, whose actions had to be coordinated and synchronized in real time. This was achieved through use of the AccessGrid[278, 279] teleconferencing system. The IRC text conferencing system[280] proved invaluable for coordination of the simulations: its text-based nature allowed the easy exchange of URLs and GSHs between participants, and the low bandwidth and processing requirements of IRC clients allow them to be left running in the background for asynchronous communication, as opposed to the AccessGrid meetings, which required specially equipped physical venues and correspondingly constrained time schedules. An editable web-page, or Wiki[252], was used as a combined bulletin board, configuration repository, scheduling tool, and distributed notebook.

B.5 Practical observations

A significant amount of effort was required to make sure that all the required software was ported to and ran smoothly on the required platforms: not only the application and visualization codes, but also the libraries on which they relied. A significant problem in using such a heterogeneous Grid is that the location and invocation of compilers and libraries differ widely, even between machines of the same architecture. Environmental parameters, such as the location of temporary and permanent filespace, file retention policies, or executable paths, also varied widely. During the project, these issues were dealt with through the use of *ad hoc* shell scripts. but this is not a satisfactory solution in general, due to the amount of work required.

The TeraGyroid testbed network was formed by federating separate Grids in the UK and US: this required each Grid to recognize users and certificates from other Grids. During the project. this was mostly dealt with by communication between the individuals involved, and by posting the IDs of the certificates which needed to

be recognized on a Wiki; however, again, this is not a scalable long-term solution, and ideally the issue would be dealt with through use of a third-party certificate management system. Given that most definitions of Grid computing require transparent operation across multiple administrative domains. it is questionable whether the term "Grid" could even be applied to the project. Management of certificates. and a Public Key Infrastructure (PKI) in general. are considered to be difficult technical problems[281] at the time of writing, although new approaches are being suggested[282, 283].

Much use was made of "dual-homed" systems, with multiple IP addresses on multiple networks. This caused problems due to the tendency of authentication systems such as SSL to confuse host identity with IP address, requiring ugly workarounds. More generally, most networking software at present assumes a homogeneous network, and delegates control of routing to much lower levels. This makes it difficult, for example, for a client process running on one host to move files between two other hosts using a specific network, in the case (as it was with the TeraGyroid project) where a high-bandwidth network has been constructed specifically for the transfer, and is to be preferred over other links.

Problems were encountered when the compute and visualization nodes were not directly connected to the Internet, but communicated through firewalls. which is a common situation on large clusters. Workarounds such as port-forwarding and process pinning were used during the project, but again do not represent good long-term solutions.

The simulation pipeline requires AccessGrid virtual venues and simulation, visualization, and storage facilities to be available simultaneously, at times when their human operators could reasonably expect to be around. This was often dealt with by manual reservation of resources by systems administrators, but the ideal solution would involve automated advance reservation and co-allocation procedures.

It was generally found that existing middleware toolkits such as Globus were rather heavyweight, requiring substantial effort and local tuning on the part of systems administrators to install and maintain, particularly due to their reliance on specific versions of custom-patched libraries. The high-level lightweight Grid Service Container `OGSI::Lite`[284] was found to be invaluable to the project, since it allowed very rapid development and hosting of high-level services such as the Registry in a way which would not have been possible on similar timescales using toolkits such as Globus. These issues are examined in closer detail in a separate document[271].

B.6 Conclusions and future work

At the time of writing, Grid computing is a very rapidly evolving field, with many approaches to its various problems being proposed. designed, built. and evaluated.

In the specific context of simulations performed over a Grid, a large part of the success of the TeraGyroid project in being able to run multiple steered simulations over a network of diverse machines subject to complicated resource constraints has been attributed[196, 260, 271] to the `OGSI::Lite` Grid Services Container. which allowed small Grid Services to be very rapidly developed and made available. While on the face of it, the job of running a large-scale calculation might seem like a single, monolithic task. in reality it requires a very large number of ancillary tasks to allocate and schedule resources, preprocess input data. move intermediate data.

and postprocess, visualize, store and curate output data. The project would have been significantly more difficult had it not had the ability to create and deploy small services to perform these tasks at relatively short notice.

This insight, along with recognition that some of the existing toolkits were too large and cumbersome to permit widespread deployment[271], has led to ongoing research initiatives[285, 286] in the field of "lightweight" Grid computing. These initiatives aim to reduce the barrier to entry of Grid computing, both in terms of the complexity and bulk of the software required, and also in terms of the concepts the end-user must first understand. It is hoped that such software will have better evolutionary characteristics[287] and correspondingly better uptake.

Bibliography

[1] T. E. Faber. *Fluid Dynamics for Physicists.* Cambridge University Press (1995). ISBN 0-521-42969-2

[2] M. Born and H. S. Green. 'A general kinetic theory of liquids. I. the molecular distribution functions'. *Proc. R. Soc. Lond. A*, 188(1012) 10–18 (1946)

[3] H. S. Green. 'A general kinetic theory of liquids. II. equilibrium properties'. *Proc. R. Soc. Lond. A*, 129(1016) 103–117 (1947)

[4] M. Born and H. S. Green. 'A general kinetic theory of liquids. III dynamical properties.' *Proc. R. Soc. Lond. A*, 190(1023) 455–474 (1947)

[5] M. Born and H. S. Green. 'A general kinetic theory of liquids. IV quantum mechanics of fluids'. *Proc. R. Soc. Lond. A*, 191(1025) 168–181 (1947)

[6] M. Born and H. S. Green. 'A general kinetic theory of liquids. V. liquid He II'. *Proc. R. Soc. Lond. A*, 194(1037) 244–258 (1948)

[7] R. L. Liboff. *Kinetic Theory: Classical, Quantum, and Relativistic Descriptions.* Prentice-Hall (1990). ISBN 0387955518

[8] J. Moore. 'Physics 202 course notes' (2003).
URL `http://socrates.berkeley.edu/%7ejemoore/p212/phys212.html`

[9] S. Chapman and T. G. Cowling. *The Mathematical Theory of Non-uniform Gases.* Cambridge University Press, second edition (1952). ISBN 0-521-40844-X

[10] P. L. Bhatnagar, E. P. Gross, and M. Krook. 'Model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems'. *Phys. Rev.*, 94(3) 511–525 (1954)

[11] S. Wolfram. 'Cellular automaton fluids 1: Basic theory'. *J. Stat. Phys.*, 45(3/4) 471–526 (1986)

[12] U. Frisch, B. Hasslacher, and Y. Pomeau. 'Lattice-gas automata for the Navier-Stokes equation'. *Phys. Rev. Lett.*, 56(14) 1505–1508 (1986)

[13] B. M. Boghosian, P. V. Coveney, and P. J. Love. 'A three dimensional lattice-gas model for amphiphilic fluid dynamics'. *Proc. R. Soc. Lond. A*. 456 1431 (2000)

[14] G. Zanetti. 'Hydrodynamics of lattice-gas automata'. *Phys. Rev. A*. 40(3) 1539–1548 (1989).
doi:10.1103/PhysRevA.40.1539

[15] U. Frisch, D. d'Humières, B. Hasslacher, P. Lallemand, Y. Pomeau, and J.-P. Rivet. 'Lattice gas hydrodynamics in two and three dimensions'. *Complex Systems*, 1 649–707 (1987)

[16] B. M. Boghosian, P. J. Love, and D. A. Meyer. 'Towards the simplest hydrodynamic lattice-gas model'. *Phil. Trans. R. Soc. Lond. A*, 360(1792) 333 – 344 (2002)

[17] G. R. McNamara and G. Zanetti. 'Use of the Boltzmann equation to simulate lattice-gas automata'. *Phys. Rev. Lett.*, 61(20) 2332–2335 (1988)

[18] D. d'Humières and P. Lallemand. 'numerical simulations of hydrodynamics with lattice gas automata in two dimensions'. *Complex Systems*, 1 599–632 (1987)

[19] P. J. Higuera, S. Succi, and R. Benzi. 'Lattice gas dynamics with enhanced collisions'. *Europhys. Lett.*, 9(4) 345–349 (1989)

[20] F. J. Higuera and J. Jiménez. 'Boltzmann approach to lattice gas simulations'. *Europhys. Lett.*, 9(7) 663–668 (1989)

[21] Y. H. Qian, D. d'Humières, and P. Lallemand. 'Lattice BGK models for Navier-Stokes equation'. *Europhys. Lett.*, 17(6) 479–484 (1992)

[22] S. Chen, H. Chen, D. Martínez, and W. Matthaeus. 'Lattice Boltzmann model for simulation of magnetohydrodynamics'. *Phys. Rev. Lett.*, 67(27) 3776–3779 (1991)

[23] H. Chen, S. Chen, and W. H. Matthaeus. 'Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method'. *Phys. Rev. A*, 45(8) 5339–5341 (1992)

[24] A. J. Wagner. 'An H-theorem for the lattice Boltzmann approach to hydrodynamics'. *Europhys. Lett.*, 44 144–149 (1998). doi:10.1209/epl/i1998-00448-8

[25] Y. H. Qian. *Gaz sur Réseax at Théorie Cinétique sur Réseaux Appliquée à l'Equation de Navier-Stokes.* Ph.D. thesis, École Normale Supérieure, 45 rue d'Ulm, 75230 Paris cedex 05 (1990)

[26] D. H. Rothman and J. M. Keller. 'Immiscible cellular-automaton fluids'. *J. Stat. Phys.*, 52(3–4) 1119–1127 (1988)

[27] J. F. Olson and D. H. Rothman. 'Two-fluid flow in sedimentary rock: simulation, transport and complexity'. *J. Fluid Mech.*, 341 343–370 (1997)

[28] X. He and L.-S. Luo. 'Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation'. *Phys. Rev. E.* 56(6) 6811–6817 (1997). doi:10.1103/PhysRevE.56.6811

[29] L.-S. Luo. 'Unified theory of lattice Boltzmann models for nonideal gases'. *Phys. Rev. Lett.*, 81 1618–1621 (1998)

[30] T. Abe. 'Derivation of the lattice Boltzmann method by means of the discrete ordinal method for the Boltzmann equation'. *J. Comp. Phys.*. 131 241–246 (1997)

[31] P. J. Love, M. Nekovee, P. V. Coveney, J. Chin, N. González-Segredo. and J. M. R. Martin. 'Simulations of amphiphilic fluids using mesoscale lattice-Boltzmann and lattice-gas methods'. *Comp. Phys. Comm.*, 153 340–358 (2003).
doi:10.1016/S0010-4655(03)00200-5

[32] P. J. Hoogerbrugge and J. M. V. A. Koelman. 'Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics'. *Europhys. Lett.*. 19(3) 155–160 (1992)

[33] P. Español and P. Warren. 'Statistical mechanics of dissipative particle dynamics'. *Europhys. Lett.*, 30(4) 191–196 (1995)

[34] P. Español. 'Hydrodynamics from dissipative particle dynamics'. *Phys. Rev. E*, 52(2) 1734–1742 (1995)

[35] L. E. Reichl. *A Modern Course in Statistical Physics.* J. Wiley and Sons, second edition (1998). ISBN 0-471-59520-9

[36] C. A. Marsh, G. Backx, and M. H. Ernst. 'Static and dynamic properties of dissipative particle dynamics'. *Phys. Rev. E*, 56(2) 1676–1691 (1997)

[37] P. V. Coveney and P. Español. 'Dissipative particle dynamics for interacting multicomponent systems'. *J. Phys. A.*, 30 779–784 (1997)

[38] C. A. Marsh and P. V. Coveney. 'Detailed balance and H-theorems for dissipative particle dynamics'. *J. Phys. A.*, 31 6561–6568 (1998)

[39] P. V. Coveney and K. E. Novik. 'Computer simulations of domain growth and phase separation in two-dimensional binary immiscible fluids using dissipative particle dynamics'. *Phys. Rev. E*, 54(5) 5134–5141 (1996).
doi:10.1103/PhysRevE.54.5134

[40] S. I. Jury. *Computer Simulation of Complex Fluids using Dissipative Particle Dynamics.* Ph.D. thesis, University of Edinburgh (1999)

[41] S. Jury, P. Bladon, M. Cates, S. Krishna, M. Hagen, N. Ruddock, and P. Warren. 'Simulation of amphiphilic mesophases using dissipative particle dynamics'. *Phys. Chem. Chem. Phys.*, 1 2051–2056 (1999)

[42] E. S. Boek, P. V. Coveney, H. N. W. Lekkerkerker, and P. van der Schoot. 'Simulating the rheology of dense colloidal suspensions using dissipative particle dynamics'. *Phys. Rev. E*, 55(3) 3124–3133 (1997)

[43] P. Español. 'A fluid particle model'. *Phys. Rev. E.* 57(3) 2390–2948 (1998)

[44] E. G. Flekkøy, P. V. Coveney, and G. D. Fabritiis. 'Foundations of dissipative particle dynamics'. *Phys. Rev. E*, 62(2) 2140–2157 (2000)

[45] A. Malevanets and R. Kapral. 'Continuous-velocity lattice-gas model for fluid flow'. *Europhys. Lett.*, 44(5) 552–558 (1998). doi:10.1209/epl/i1998-00508-7

[46] Y. Hashimoto, Y. Chen, and H. Ohashi. 'Immiscible real-coded lattice gas'. *Comp. Phys. Comm.*, 129(1–3) 56–62 (2000). doi:10.1016/S0010-4655(00)00092-8

[47] T. Sakai, Y. Chen, and H. Ohashi. 'Formation of micelle in the real-coded lattice gas'. *Comp. Phys. Comm.*, 129(1–3) 75–81 (2000). doi:10.1016/S0010-4655(00)00094-1

[48] E. Tüzel, M. Strauss, T. Ihle, and D. M. Kroll. 'Transport coefficients for stochastic rotation dynamics in three dimensions'. *Phys. Rev. E*, 68 036701 (2003). doi:10.1103/PhysRevE.68.036701

[49] A. Malevanets and R. Kapral. 'Mesoscopic model for solvent dynamics'. *J. Chem. Phys.*, 110(17) 8605–8613 (1999)

[50] A. Malevanets and J. M. Yeomans. 'Dynamics of short polymer chains in solution'. *Europhys. Lett.*, 52(2) 231–237 (2000). doi:10.1209/epl/i2000-00428-0

[51] R. Delgado-Buscalioni and P. V. Coveney. 'Continuum-particle hybrid coupling for mass, momentum, and energy transfers in unsteady fluid flow'. *Phys. Rev. E*, 67(046704) (2003). doi:10.1103/PhysRevE.67.046704

[52] A. J. C. Ladd. 'Short-time motion of colloidal particles: Numerical simulation via a fluctuating lattice-Boltzmann equation'. *Phys. Rev. Lett.*, 70(9) 1339–1342 (1993)

[53] M. E. Cates, K. Stratford, R. Adhikari, P. Stansell, J. C. Desplat, I. Pagonabarraga, and A. J. Wagner. 'Simulating colloid hydrodynamics with lattice Boltzmann methods'. *J. Phys. Condens. Matter*, 16(38) S3903–S3915 (2004). doi:10.1088/0953-8984/16/38/009

[54] A. K. Gunstensen, D. H. Rothman, S. Zaleski, and G. Zanetti. 'Lattice Boltzmann model of immiscible fluids'. *Phys. Rev. A*, 43(8) 4320–4327 (1991)

[55] S. Hou, X. Shan, Q. Zou, G. Doolen, and W. Soll. 'Evaluation of two lattice Boltzmann models for multiphase flows'. *J. Comp. Phys.*, 138 695–713 (1997). doi:10.1006/jcph.1997.5839

[56] D. Grunau, S. Chen, and K. Eggert. 'A lattice Boltzmann model for multiphase fluid flows'. *Phys. Fluids A*, 5(10) 2557–2651 (1993)

[57] X. Shan and H. Chen. 'Lattice Boltzmann model for simulating flows with multiple phases and components'. *Phys. Rev. E.* 47(3) 1815–1819 (1993). doi:10.1103/PhysRevE.47.1815

[58] X. Shan and H. Chen. 'Simulation of nonideal gases and liquid-gas phase transitions by the lattice Boltzmann equation'. *Phys. Rev. E.* 49(4) 2941–2948 (1994).
doi:10.1103/PhysRevE.49.2941

[59] M. R. Swift, W. R. Osborn, and J. M. Yeomans. 'Lattice Boltzmann simulation of nonideal fluids'. *Phys. Rev. E*, 75(5) 830–833 (1995)

[60] E. Orlandini, M. R. Swift, and J. M. Yeomans. 'A lattice Boltzmann model of binary-fluid mixtures'. *Europhys. Lett.*, 32(6) 463–468 (1995)

[61] M. R. Swift, E. Orlandini, W. R. Osborn, and J. M. Yeomans. 'Lattice-Boltzmann simulations of liquid-gas and binary fluid mixtures'. *Phys. Rev. E.* 54(5) 5041–5052 (1996)

[62] A. N. Kalarakis, V. N. Burganos, and A. C. Payatakes. 'Galilean-invariant lattice-Boltzmann simulation of liquid-vapor interface dynamics'. *Phys. Rev. E*, 65 056702 (2002).
doi:10.1103/PhysRevE.65.056702

[63] L.-S. Luo. 'Theory of the lattice Boltzmann method: Lattice Boltzmann models for nonideal gases'. *Phys. Rev. E*, 62(4) 4982–4995 (2000).
doi:10.1103/PhysRevE.62.4982

[64] P. Lallemand and L.-S. Luo. 'Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability'. *Phys. Rev. E*, 61(6) 6546 – 6562 (2000).
doi:10.1103/PhysRevE.61.6546

[65] D. d'Humières, I. Ginzburg, M. Krafczyk, P. Lallemand, and L.-S. Luo. 'Multiple-relaxation-time lattice Boltzmann models in three dimensions'. *Phil. Trans. R. Soc. Lond. A*, 360(1792) 437 – 451 (2002).
doi:10.1098/rsta.2001.0955

[66] L.-S. Luo and S. S. Girimaji. 'Lattice Boltzmann model for binary mixtures'. *Phys. Rev. E*, 66(035301) (2002).
doi:10.1103/PhysRevE.66.035301

[67] B. M. Boghosian and P. V. Coveney. 'Inverse Chapman-Enskog derivation of the thermohydrodynamic lattice-BGK model'. *Int. J. Mod. Phys. C*, 9 1231–1246 (1998).
URL `http://xxx.lanl.gov/abs/comp-gas/9810001`

[68] H. Chen, B. M. Boghosian, P. V. Coveney, and M. Nekovee. 'A ternary lattice Boltzmann model for amphiphilic fluids'. *Proc. R. Soc. Lond. A.* 456 2043–2047 (2000)

[69] P. J. Love. 'A particulate basis for a lattice-gas model of amphiphilic fluids'. *Phil. Trans. R. Soc. Lond. A*, 360 345 (2002)

[70] G. Gompper and M. Schick. 'Self-assembling amphiphilic systems'. In C. Domb and J. Lebowitz, editors, 'Phase Transitions and Critical Phenomena', volume 16, pages 1–176. Academic Press (1994)

[71] J. Chin, J. Harting, S. Jha, P. V. Coveney, A. R. Porter, and S. M. Pickles. 'Steering in computational science: mesoscale modelling and simulation'. *Contemporary Physics*, 44(5) 417–434 (2003). doi:10.1080/0010751031000160504 6

[72] A. J. Bray. 'Theory of phase-ordering kinetics'. *Adv. Phys.*, 43(3) 357–459 (1994)

[73] H. Furukawa. 'A dynamic scaling assumption for phase separation'. *Adv. Phys.*, 34(6) 703–750 (1985)

[74] M. Grant and K. R. Elder. 'Spinodal decomposition in fluids'. *Phys. Rev. Lett.*, 82(1) 14–16 (1999)

[75] J. W. Cahn and J. E. Hilliard. 'Free energy of nonuniform system. I. interfacial free energy'. *J. Chem. Phys.*, 28(2) 258–267 (1958)

[76] J. D. Gunton, M. S. Miguel, and P. S. Sahni. 'The dynamics of first order phase transitions'. In C. Domb and J. Lebowitz, editors, 'Phase Transitions and Critical Phenomena', volume 8, pages 319–339. Academic Press New York, 15 East 26th Street, NY 10010 (1983)

[77] Y. Wu, F. J. Alexander, T. Lookman, and S. Chen. 'effects of hydrodynamics on phase transition kinetics in two-dimensional binary fluids'. *Phys. Rev. Lett.*, 74(19) 3852–3855 (1995)

[78] J. E. Farrell and O. T. Valls. 'Spinodal decomposition in a two-dimensional fluid model'. *Phys. Rev. B*, 40(10) 7027–7039 (1989). doi:10.1103/PhysRevB.40.7027

[79] M. E. Cates, V. M. Kendon, P. Bladon, and J. C. Desplat. 'Inertia, coarsening and fluid motion in binary mixtures'. *Faraday Disc.*, 112 1–11 (1999)

[80] C. Roland and M. Grant. 'Monte Carlo renormalization-group study of spinodal decomposition: Scaling and growth'. *Phys. Rev. B*, 39(16) 11971–11981 (1989)

[81] A. J. Wagner and J. M. Yeomans. 'Breakdown of scale-invariance in the coarsening of phase-separating binary fluids'. *Phys. Rev. Lett.*, 80(7) 1429–1432 (1998)

[82] R. B. Rybka, M. Cieplak, and D. Salin. 'Boltzmann cellular automata studies of the spinodal decomposition'. *Physica A*, 222 105–118 (1995)

[83] E. D. Siggia. 'Late stages of spinodal decomposition in binary mixtures'. *Phys. Rev. A*, 20(2) 595–605 (1979)

[84] M. San Miguel, M. Grant, and J. D. Gunton. 'Phase separation in two-dimensional binary fluids'. *Phys. Rev. A*, 31(2) 1001–1005 (1985)

[85] P. B. Sunil Kumar and M. Rao. 'Novel Monte Carlo approach to the dynamics of fluids: Single-particle diffusion, correlation functions, and phase ordering of binary fluids'. *Phys. Rev. Lett.*, 77(6) 1067–1070 (1996). doi:10.1103/PhysRevLett.77.1067

[86] E. Velasco and S. Toxvaerd. 'Computer simulation of phase separation in a two-dimensional binary fluid mixture'. *Phys. Rev. Lett.*. 71(3) 388–391 (1993). doi:10.1103/PhysRevLett.71.388

[87] K. E. Novik and P. V. Coveney. 'Spinodal decomposition of off-critical quenches with a viscous phase using dissipative particle dynamics in two and three dimensions'. *Phys. Rev. E*, 61(1) 435–448 (2000)

[88] A. N. Emerton, P. V. Coveney, and B. M. Boghosian. 'Lattice-gas simulations of domain growth, saturation and self-assembly in immiscible fluids and microemulsions'. *Phys. Rev. E*, 56(1) 1286–1306 (1997)

[89] D. W. Grunau, T. Lookman, S. Y. Chen, and A. S. Lapedes. 'Domain growth. wetting, and scaling in porous media'. *Phys. Rev. Lett.*. 71(25) 4198–4201 (1993)

[90] W. R. Osborn, E. Orlandini, M. R. Swift, J. M. Yeomans, and J. R. Banavar. 'Lattice Boltzmann study of hydrodynamic spinodal decomposition'. *Phys. Rev. Lett.*, 75(22) 4031–4034 (1995)

[91] H. Chen and A. Chakrabarti. 'Surface-directed spinodal decomposition: Hydrodynamic effects'. *Phys. Rev. E*, 55(5) 5680–5688 (1997)

[92] S. Chen and T. Lookman. 'Growth kinetics in multicomponent fluids'. *J. Stat. Phys.*, 81 223–235 (1995)

[93] F. J. Alexander, S. Chen, and D. W. Grunau. 'Hydrodynamic spinodal decomposition: Growth kinetics and scaling functions'. *Phys. Rev. B*, 48(1) 634–637 (1993)

[94] V. M. Kendon, J. C. Desplat, P. Bladon, and M. E. Cates. '3D spinodal decomposition in the inertial regime'. *Phys. Rev. Lett.*, 83(3) 576–579 (1999). doi:10.1103/PhysRevLett.83.576

[95] H. Tanaka. 'Double phase separation in a confined symmetric binary mixture: Interface quench effect unique to bicontinuous phase separation'. *Phys. Rev. Lett.*, 72(23) 3690–3693 (1994)

[96] V. M. Kendon, M. E. Cates, I. Pagonabarraga, J. C. Desplat, and P. Bladon. 'Inertial effects in three-dimensional spinodal decomposition of a symmetric binary fluid mixture: A lattice Boltzmann study'. *J. Fluid Mech.*, 440 147–203 (2001)

[97] J. Walker. 'The amateur scientist: Fluid interfaces, including fractal flows, can be studied in a Hele-Shaw cell'. *Scientific American*, 257 134 138 (1997)

[98] P. G. Saffman and G. Taylor. 'The penetration of a fluid into a porous medium or Hele-Shaw cell containing a more viscous liquid'. *Proc. R. Soc. Lond. A*, 245 312–329 (1958)

[99] H. J. S. Hele-Shaw. 'The flow of water'. *Nature*. 58(1489) 34–36 (1898)

[100] E. G. Flekkøy, U. Oxaal, J. Feder, and T. Jøssang. 'Hydrodynamic dispersion at stagnation points: Simulations and experiments'. *Phys. Rev. E.* 52(5) 4952–4962 (1995).
doi:10.1103/PhysRevE.52.4952

[101] E. G. Flekkøy. 'Lattice Bhatnagar-Gross-Krook models for miscible fluids'. *Phys. Rev. E*, 47(6) 4247–4257 (1993)

[102] R. Bird, W. E. Stewart, and E. N. Lightfoot. *Transport Phenomena.* J. Wiley and Sons (1965). ISBN 0-471-07392-X

[103] P. Gondret, N. Rakotomalala, M. Rabaud, D. Salin, and P. Watzky. 'Viscous parallel flows in finite aspect ratio Hele-Shaw cell: Analytical and numerical results'. *Phys. Fluids*, 9(6) 1841–1843 (1997).
doi:10.1063/1.869301

[104] E. G. Flekkøy, T. Rage, U. Oxaal, and J. Feder. 'Hydrodynamic irreversibility in creeping flow'. *Phys. Rev. Lett.*, 77(20) 4170–4173 (1996).
doi:10.1103/PhysRevLett.77.4170

[105] P. Grosfils and J.-P. Boon. 'Viscous fingering in miscible, immiscible and reactive fluids'. *Int. J. Mod. Phys. B*, 17(1–2) 15–20 (2003).
doi:10.1142/S0217979203017023

[106] P. Grosfils, J. P. Boon, J. Chin, and E. S. Boek. 'Structural and dynamical characterization of Hele-Shaw viscous fingering'. *Phil. Trans.: Mat. Phys. Eng. Sci.*, 362(1821) 1471–2962 (2004).
doi:10.1098/rsta.2004.1398

[107] K. Langaas and J. M. Yeomans. 'Lattice Boltzmann simulation of a binary fluid with different phase viscosities and its application to fingering in two dimensions'. *Eur. Phys. J. B*, 15(1) 133–141 (1999)

[108] J. Chin, E. S. Boek, and P. V. Coveney. 'Lattice Boltzmann simulation of the flow of binary immiscible fluids with different viscosities using the Shan-Chen microscopic interaction model'. *Phil. Trans.: Mat. Phys. Eng. Sci.*, 360(1792) 547–558 (2002).
doi:10.1098/rsta.2001.0953

[109] N. S. Martys and J. F. Douglas. 'Critical properties and phase separation in lattice Boltzmann fluid mixtures'. *Phys. Rev. E.* 63 031205 (2001).
doi:10.1103/PhysRevE.63.031205

[110] C. T. Tan and G. M. Homsy. 'Stability of miscible displacements in porous media: Rectilinear flow'. *Phys. Fluids*, 29(11) 3549–3556 (1986).
doi:10.1063/1.865832

[111] S. Hill. 'Channeling in packed columns'. *Chem. Eng. Sci.*, 1(6) 247–253 (1952).
doi:10.1016/0009-2509(52)87017-4

[112] G. M. Homsy. 'Viscous fingering in porous media'. *Annu. Rev. Fluid Mech.*, 19 271–311 (1987).
doi:10.1146/annurev.fl.19.010187.001415

[113] D. Bensimon, L. P. Kadanoff, S. Liang, B. I. Shraiman, and C. Tang. 'Viscous flows in two dimensions'. *Rev. Mod. Phys.*, 58(4) 977–999 (1986). doi:10.1103/RevModPhys.58.977

[114] S. Tanveer. 'Surprises in viscous fingering'. *J. Fluid Mech.*, 409 273–308 (2000)

[115] W. B. Zimmerman and G. M. Homsy. 'Nonlinear viscous fingering in miscible displacement with anisotropic dispersion'. *Phys. Fluids A*, 3(8) 1859–1872 (1991). doi:10.1063/1.857916

[116] W. B. Zimmerman and G. M. Homsy. 'Three-dimensional viscous fingering: A numerical study'. *Phys. Fluids A*, 4(9) 1901–1914 (1992). doi:10.1063/1.858361

[117] W. B. Zimmerman and G. M. Homsy. 'Viscous fingering in miscible displacements: Unification of effects of viscosity contrast, anisotropic dispersion, and velocity dependence of dispersion on nonlinear finger propagation'. *Phys. Fluids A*, 4(11) 2348–2359 (1992). doi:10.1063/1.858476

[118] X. Guan and R. Pitchumani. 'Viscous fingering in a Hele-Shaw cell with finite viscosity ratio and interfacial tension'. *J. Fluid Eng.*, 125(2) 354–364 (2003). doi:10.1115/1.1524589

[119] R. Folch, J. Casademunt, A. Hernández-Machado, and L. Ramírez-Piscina. 'Phase-field model for Hele-Shaw flows with arbitrary viscosity contrast. I. theoretical approach'. *Phys. Rev. E*, 60(2) 1724–1733 (1999). doi:10.1103/PhysRevE.60.1724

[120] R. Folch, J. Casademunt, A. Hernández-Machado, and L. Ramírez-Piscina. 'Phase-field model for Hele-Shaw flows with arbitrary viscosity contrast. II. numerical study'. *Phys. Rev. E*, 60(2) 1734–1740 (1999). doi:10.1103/PhysRevE.60.1734

[121] C. Tang. 'Diffusion-limited aggregation and the Saffman-Taylor problem'. *Phys. Rev. A*, 31(3) 1977–1979 (1985). doi:10.1103/PhysRevA.31.1977

[122] V. A. Bogoyavlenskiy. 'Bridge from diffusion-limited aggregation to the Saffman-Taylor problem'. *Phys. Rev. E*, 63(4) 045305 (2001). doi:10.1103/PhysRevE.63.045305

[123] D. Burgess and F. Hayot. 'Saffman-Taylor-type instability in a lattice gas'. *Phys. Rev. A*, 40(9) 5187–5192 (1989). doi:10.1103/PhysRevA.40.5187

[124] N. Rakotomalala, D. Salin, and P. Watzky. 'Miscible displacement between two parallel plates: BGK lattice gas simulations'. *J. Fluid Mech.*, 338 277–297 (1997)

[125] D. A. Vasquez and A. D. Wit. 'Dispersion relations for the convective instability of an acidity front in Hele-Shaw cells'. *J. Comp. Phys.*, 121(2) 935–941

(2004).
doi:10.1063/1.1760515

[126] A. de Wit. 'Miscible density fingering of chemical fronts in porous media: Nonlinear simulations'. *Phys. Fluids*, 16(1) 163–175 (2004).
doi:10.1063/1.1630576

[127] T. Bánsági, D. Horváth, and Á. Tóth. 'Convective instability of an acidity front in Hele-Shaw cells'. *Phys. Rev. E*, 68 026303 (2003).
doi:10.1103/PhysRevE.68.026303

[128] S. H. Kang, H. G. Im, and S. W. Baek. 'A computational study of Saffman-Taylor instability in premixed flames'. *Combust. Theory Modelling.* 7(2) 343–363 (2003).
doi:10.1088/1364-7830/7/2/308

[129] O. Zik, Z. olami, and E. Moses. 'Fingering instability in combustion'. *Phys. Rev. Lett.*, 81(18) 3868–3871 (1998).
doi:10.1103/PhysRevLett.81.3868

[130] H. Zhao and J. V. Maher. 'Viscous-fingering experiments with periodic boundary conditions'. *Phys. Rev. A*, 42(10) 5894–5897 (1990).
doi:10.1103/PhysRevA.42.5894

[131] J. V. Maher. 'Development of viscous fingering patterns'. *Phys. Rev. Lett.*, 54(14) 1498–1501 (1985).
doi:10.1103/PhysRevLett.54.1498

[132] M. W. DiFrancesco and J. V. Maher. 'Noisy and regular features in Saffman-Taylor patterns'. *Phys. Rev. A*, 39(9) 4709–4717 (1989).
doi:10.1103/PhysRevA.39.4709

[133] G. Tryggvason and H. Aref. 'Numerical experiments on Hele-Shaw flow with a sharp interface'. *J. Fluid Mech.*, 136 1–30 (1983)

[134] G. Tryggvason and H. Aref. 'Finger-interaction mechanisms in stratified Hele-Shaw flow'. *J. Fluid Mech.*, 154 287–301 (1985)

[135] M. M. Dupin, I. Halliday, and C. M. Care. 'Multi-component lattice Boltzmann equation for mesoscale blood flow'. *J. Phys. A.*, 36 8517–8534 (2003).
doi:10.1088/0305-4470/36/31/313

[136] X. D. Niu, C. Shu, and Y. T. Chew. 'A lattice Boltzmann BGK model for simulation of micro flows'. *Europhys. Lett.*, 67(4) 600–606 (2004).
doi:10.1209/epl/i2003-10307-8

[137] D. Langevin. 'Surfactant systems'. *Europhysics News.* pages 72–73 (1999)

[138] J. M. Seddon and R. H. Templer. *Polymorphism of Lipid-Water Systems*, chapter 3, pages 97–160. Elsevier Science B. V. (1995)

[139] J. M. Seddon and R. H. Templer. 'Cubic phases of self-assembled amphiphilic aggregates'. *Phil. Trans.: Phys. Sci. Eng.*. 344(1672) 377–401 (1993)

[140] International Union of Pure and Applied Chemistry. *IUPAC Compendium of Chemical Terminology*. IUPAC, second edition (1997).
URL `http://www.chemsoc.org/chembytes/goldbook/`

[141] P. B. Canham. 'The minimum energy of bending as a possible explanation of the biconcave shape of the human red blood cell'. *J. Theoret. Biol.*, 26 61–81 (1970)

[142] G. E. Schröder, S. J. Ramsden, A. Fogden, and S. T. Hyde. 'A rhombohedral family of minimal surfaces as a pathway between the P and D cubic mesophases'. *Physica A*, 339(1–2) 137–144 (2004)

[143] L. E. Scriven. 'Equilibrium bicontinuous structure'. *Nature*, 263 123–125 (1976)

[144] A. L. Mackay. 'Periodic minimal surfaces'. *Nature*, 314 604–606 (1985)

[145] R. Courant and D. Hilbert. *Methods of Mathematical Physics Volume 1*. J. Wiley and Sons (1953). ISBN 0-470-17952-X

[146] H. Flanders. *Differential Forms with Applications to the Physical Sciences*. Academic Press (1963). ISBN 0-486-66169-5

[147] U. Schwarz and G. Gompper. *Biocontinuous Surfaces in Self-assembling Amphiphilic Systems*, volume 600 of *Springer Lecture Notes in Physics*, pages 107–151. Springer (2002)

[148] A. Schoen. 'Infinite periodic minimal surfaces without self-intersections'. *NASA Tech. Note*, D-5541 (Washington, DC, 1970)

[149] H. Karcher and K. Polthier. 'Construction of triply periodic minimal surfaces'. *Phil. Trans. R. Soc. Lond. A*, 354(1715) 2077–2104 (1996)

[150] E. A. Lord and A. L. Mackay. 'Periodic minimal surfaces of cubic symmetry'. *Cur. Sci.*, 85(3) 346–362 (2003).
URL `http://www.ias.ac.in/currsci/aug102003/346.pdf`

[151] J. Klinowski, A. L. Mackay, and H. Terrones. 'Curved surfaces in chemical structure'. *Proc. R. Soc. Lond. A*, 364(1715) 1975–1987 (1996)

[152] W. Góźdź and R. Hołyst. 'High genus periodic gyroid surfaces of nonpositive gaussian curvature'. *Phys. Rev. Lett.*, 76(15) 2726–2729 (1996).
doi:10.1103/PhysRevLett.76.2726

[153] S. T. Hyde and S. Ramsden. 'Polycontinuous morphologies and interwoven helical networks'. *Europhys. Lett.*, 50(2) 135–141 (2000).
doi:10.1209/epl/i2000-00245-y

[154] J. D. Enlow, R. L. Enlow, K. M. McGrath, and M. W. Tate. 'Modeling liquid crystal bilayer structures with minimal surfaces'. *J. Comp. Phys.*, 120(4) 1981–1989 (2004)

[155] S. T. Hyde and G. E. Schroeder. 'Novel surfactant mesostructural topologies: between lamellae and columnar (hexagonal) forms'. *Cur. Opin. Colloid Int. Sci.*, 8(1) 5–14 (2003)

[156] S. T. Hyde. 'Microstructure of bicontinuous surfactant aggregates'. *J. Phys. Chem.*, 93(4) 1458–1464 (1989)

[157] A. Fogden and S. T. Hyde. 'Continuous transformations of cubic minimal surfaces'. *Eur. Phys. J. B*, 7(1) 91–104 (1999)

[158] P. J. F. Gandy and J. Klinowski. 'Exact computation of the triply periodic G ('gyroid') minimal surface'. *Chem. Phys. Lett.*, 321(5) 363–371 (2000)

[159] K. Große-Brauckmann. 'Gyroids of constant mean curvature'. *Exp. Math.*, 6(1) 33–50 (1997).
URL `http://www.expmath.org/restricted/6/6.1/gyr.ps`

[160] U. S. Schwarz and G. Gompper. 'Stability of inverse bicontinuous cubic phases in lipid-water mixtures'. *Phys. Rev. Lett.*, 85(7) 1472–1475 (2000).
doi:10.1103/PhysRevLett.85.1472

[161] A. Avgeropoulos, B. J. Dair, N. Hadjichristidis, and E. L. Thomas. 'Tricontinuous double gyroid cubic phase in triblock copolymers of the ABA type'. *Macromolecules*, 30(19) 5634–5642 (1997).
doi:10.1021/ma970266z

[162] T. A. Shefelbine, M. E. Vigild, M. W. Matsen, D. A. Hajduk, M. A. Hillmyer, E. L. Cussler, and F. S. Bates. 'Core-shell gyroid morphology in a poly(isoprene-block-styrene-block-dimethylsiloxane) triblock copolymer'. *J. Am. Chem. Soc.*, 121(37) 8457–8465 (1999)

[163] C. Czeslik and R. Winter. 'Structure of water confined in the gyroid cubic phase of the lipid monoelaidin'. *J. Mol. Liq.*, 98 283–291 (2002)

[164] P. Mariani, B. Paci, P. Bösecke, C. Ferrero, M. Lorenzen, and R. Caciuffo. 'Effects of hydrostatic pressure on the monoolein-water system: An estimate of the energy function of the inverted Ia3d cubic phase'. *Phys. Rev. E*, 54(5) 5840–5843 (1996)

[165] V. Z. H. Chan, J. Hoffman, V. Y. Lee, H. Iatrou, A. Avgeropoulos, N. Hadjichristidis, R. D. Miller, and E. L. Thomas. 'Ordered bicontinuous nanoporous and nanorelief ceramic films from self assembling polymer precursors'. *Science*, 286(5445) 1716–1719 (1999)

[166] A. M. Urbas, M. Maldovan, P. DeRege, and E. L. Thomas. 'Bicontinuous cubic block copolymer photonic crystals'. *Adv. Mater.*, 14(24) 1850–1853 (2002)

[167] J. S. Patton and M. C. Carey. 'Watching fat digestion'. *Science*, 204(4389) 145–148 (1979)

[168] M. W. Rigler, R. E. Honkanen, and J. S. Patton. 'Visualization by freeze fracture, in vitro and in vivo, of the products of fat digestion'. *J. Lipid Res.*, 27 836–857 (1986)

[169] J. H. Prestegard and M. P. O'Brien. 'Membrane and vesicle fusion'. *Ann. Rev. Phys. Chem.*, 38 383–411 (1987)

[170] R. P. Rand. 'Interacting phospholipid bilayers: Measured forces and induced structural changes'. *Ann. Rev. Biophys. Bioeng.*. 10 277–314 (1981)

[171] A. Stier, S. A. E. Finch, and B. Bösterling. 'Non-lamellar structure in rabbit liver microsomal membranes'. *FEBS Lett.*. 91(1) (1978)

[172] G. Donnay and D. L. Pawson. 'X-ray diffraction studies of echinoderm plates'. *Science*, 166(3909) 1147–1150 (1969)

[173] H.-U. Nissen. 'Crystal orientation and plate structure in echinoid skeletal units'. *Science*, 166(3909) 1150–1152 (1969)

[174] T. Landh. 'From entangled membranes to eclectic morphologies: cubic membranes as subcellular space organizers'. *FEBS Lett.*, 369(1) 13–17 (1995). doi:10.1016/0014-5793(95)00660-2

[175] U. S. Schwarz and G. Gompper. 'Systematic approach to bicontinuous cubic phases in ternary amphiphilic systems'. *Phys. Rev. E*, 59(5) 5528–5541 (1999). doi:10.1103/PhysRevE.59.5528

[176] Wolfram Research. 'Mathworld' (2004).
URL `http://mathworld.wolfram.com/`

[177] S. T. Hyde. 'Bicontinuous structures in lyotropic liquid crystals and crystalline hyperbolic surfaces'. *Curr. Opin. Solid State and Mater. Sci.*, 1 653–662 (1996)

[178] D. Anderson, H. Wennerström, and U. Olsson. 'Isotropic bicontinuous solutions in surfactant-solvent systems: The L_3 phase'. *J. Phys. Chem.*, 93 4243–4253 (1989)

[179] R. D. Groot and T. J. Madden. 'Dynamic simulation of diblock copolymer microphase separation'. *J. Comp. Phys.*, 108(20) 8713–8724 (1998). doi:10.1063/1.476300

[180] M. Nekovee and P. Coveney. 'Lattice-Boltzmann simulations of self-assembly of a binary water-surfactant system into ordered bicontinuous cubic and lamellar phases'. *J. Am. Chem. Soc.*, 123(49) 12380–12382 (2001)

[181] N. González-Segredo, M. Nekovee, and P. V. Coveney. 'Three-dimensional lattice-Boltzmann simulations of critical spinodal decomposition in binary immiscible fluids'. *Phys. Rev. E*, 67(046304) (2003). doi:10.1103/PhysRevE.67.046304

[182] N. González-Segredo and P. V. Coveney. 'Self-assembly of the gyroid cubic mesophase: lattice-Boltzmann simulations'. *Europhys. Lett.*. 65(6) 795–801 (2004). doi:10.1209/epl/i2003-10129-8

[183] N. González-Segredo and P. V. Coveney. 'Coarsening dynamics of ternary amphiphilic fluids and the self-assembly of the gyroid and sponge mesophases: lattice-Boltzmann simulations'. *Phys. Rev. E.* 69(061501) (2004). doi:10.1103/PhysRevE.69.061501

[184] P. Prinsen, P. B. Warren, and M. A. J. Michels. 'Mesoscale simulations of surfactant dissolution and mesophase formation'. *Phys. Rev. Lett.*, 89(14) 148302 (2002).
doi:10.1103/PhysRevLett.89.148302

[185] L. E. Scriven and C. V. Sternling. 'The Marangoni effects'. *Nature*, 187 186–188 (1960)

[186] J. B. Grotberg and D. P. Gaver. 'A synopsis of surfactant spreading research'. *J. Colloid Int. Sci*, 178 377–378 (1996)

[187] J. B. Fournier and A. M. Cazabat. 'Tears of wine'. *Europhys. Lett.*, 20(6) 517–522 (1992)

[188] O. Theissen, G. Gompper, and D. M. Kroll. 'Lattice-Boltzmann model of amphiphilic systems'. *Europhys. Lett.*, 42 419–414 (1998).
doi:10.1209/epl/i1998-00265-7

[189] G. Gompper and M. Schick. 'Correlation between structural and interfacial properties of amphiphilic systems'. *Phys. Rev. Lett.*, 65(9) 1116–1119 (1990).
doi:10.1103/PhysRevLett.65.1116

[190] A. Lamura, G. Gonnella, and J. M. Yeomans. 'A lattice-Boltzmann model of ternary fluid mixtures'. *Europhys. Lett.*, 45(3) 314–320 (1999).
doi:10.1209/epl/i1999-00165-4

[191] J. Chin and P. V. Coveney. 'Lattice Boltzmann study of spinodal decomposition in two dimensions'. *Phys. Rev. E*, 66(016303) (2002).
doi:10.1103/PhysRevE.66.016303

[192] A. J. Wagner. 'The origin of spurious velocities in lattice Boltzmann'. *Int. J. Mod. Phys. B*, 17(1–2) 193–196 (2003).
doi:10.1142/S0217979203017448

[193] D. A. Hajduk, P. E. Harper, S. M. Gruner, C. C. Honeker, G. Kim, E. L. Thomas, and L. J. Fetters. 'The gyroid: A new equilibrium morphology in weakly segregated diblock copolymers'. *Macromolecules*, 27 4063 (1994)

[194] R. P. Feynman, R. B. Leighton, and M. Sands. *The Feynman Lectures on Physics*, volume 2. Addison-Wesley (1964). ISBN 0-201-02118-8

[195] C. Kittel. *Introduction to Solid State Physics*. J. Wiley and Sons, 7th edition (1996). ISBN 0-471-11181-3

[196] S. M. Pickles, R. J. Blake, B. M. Boghosian, J. M. Brooke, J. Chin, P. E. L. Clarke, P. V. Coveney, R. Haines, J. Harting, M. Harvey, S. Jha, M. A. S. Jones, M. McKeown, R. K. Pinning, A. R. Porter, K. Roy, and M. Riding. 'The TeraGyroid experiment'. *Proceedings of the Workshop on Case Studies on Grid Applications at GGF 10* (2004).
URL `http://www.realitygrid.org/TeraGyroid-Case-Study-GGF10.pdf`

[197] J. Harting, M. J. Harvey, J. Chin, and P. V. Coveney. 'Detection and tracking of defects in the gyroid mesophase'. *Comp. Phys. Comm.*, 165(2) 97–109 (2005).
doi:10.1016/j.cpc.2004.10.001

[198] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press, second edition (1992). ISBN 0-521-43108-5

[199] P. Bourke. 'Paul Bourke – personal pages'.
URL `http://astronomy.swin.edu.au/%7epbourke/`

[200] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley, second edition (1991). ISBN 0-201-12110-7

[201] W. E. Lorensen and H. E. Cline. 'Marching cubes: a high resolution 3D surface construction algorithm'. *Computer Graphics*, 21(4) 163–169 (1987). doi:10.1145/37401.37422

[202] B. P. Carneiro, C. T. Silva, and A. E. Kaufman. 'Tetra-cubes: An algorithm to generate 3D isosurfaces based upon tetrahedra'. In 'Anais do IX SIBGRAPI'. pages 205–210 (1996)

[203] G. M. Treece, R. W. Prager, and A. H. Gee. 'Regularised marching tetraheda: improved iso-surface extraction'. *Computers and Graphics*, 23 583–598 (1999). doi:10.1016/S0097-8493(99)00076-X

[204] T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares. 'Efficient implementation of marching cubes cases with topological guarantees'. *J. Graphics Tools*, 8(2) 1–15 (2003).
URL `http://www.acm.org/jgt/papers/LewinerEtAl03/`

[205] H. E. Cline, W. E. Lorensen, S. Ludke, C. R. Crawford, and B. C. Teeter. 'Two algorithms for the three-dimensional reconstruction of tomograms'. *Med. Phys.*, 15(3) 289–424 (1988)

[206] E. V. Chernyaev. 'Marching cubes 33: Construction of topologically correct isosurfaces'. Cern technical report cn 95-17, CERN (1995).
URL `http://wwwasdoc.web.cern.ch/wwwasdoc/psdir/mc.ps.gz`

[207] L. C. Kinsey. *Topology of Surfaces.* Springer-Verlag (1993). ISBN 0387941029

[208] M. A. Armstrong. *Basic Topology.* Springer-Verlag (1997). ISBN 0387908390

[209] M. Nakahara. *Geometry, Topology, and Physics.* IOP Publishing, second edition (1993). ISBN 0852740956

[210] R. Hołyst and P. Oswald. 'Confined complex liquids: Passages, droplets, permanent deformations, and order-disorder transitions'. *J. Comp. Phys.*, 109(24) 11051–11060 (1998)

[211] J. Goos and G. Gompper. 'Topological defects in lamellar phases: passages. and their fluctuations'. *J. Phys. I France*, 3 1551–1567 (1993)

[212] B. M. Boghosian, P. V. Coveney, and A. N. Emerton. 'A lattice-gas model of microemulsions'. *Proc. R. Soc. Lond. A*, 452(1948) 1221–1250 (1996)

[213] G. Gerig, T. Koller, G. Székely, C. Brechbüler, and O. Kübler. 'Symbolic description of 3-d structures applied to cerebral vessel tree obtained from MR angiography volume data'. In H. H. Barrett and A. F. Gmitro, editors. 'Lecture Notes in Computer Science No. 687, Information Processing in Medical Imaging IPMI '93', pages 94–111. Springer-Verlag (1999)

[214] S. Bischoff and L. P. Kobbelt. 'Isosurface reconstruction with topology control'. In 'Proceedings of Pacific Graphics', pages 245–255 (2002).
URL http://www-i8.informatik.rwth-aachen.de/publications/downloads/topo.

[215] K. Michielsen, H. D. Raedt, and T. Kawakatsu. 'Morphological image analysis'. In D. P. Landau, K. K. Mon, and H.-B. Schuttler, editors. 'Computer Simulation Studies in Condensed Matter Physics XII', pages 87–91. Springer-Verlag (2000)

[216] K. Michielsen, H. D. Raedt, and J. G. E. M. Fraine. 'Morphological characterization of spatial patterns'. *Prog. Theor. Phys. Suppl.*, 138 543–548 (2000)

[217] J. H. Laurer, D. A. Hajduk, J. C. Fung, J. W. Sedat, S. F. Smith, S. M. Gruner, D. A. Agard, and R. J. Spontak. 'Microstructural analysis of a cubic bicontinuous morphology in a neat SIS triblock copolymer'. *Macromolecules*. 30 3938 (1997).
doi:10.1021/ma970449l

[218] S. Sakurai, H. Umeda, C. Furukawa, H. Irie, S. Nomura, H. H. Lee, and J. K. Jim. 'Thermally induced morphological transition from lamella to gyroid in a binary blend of diblock copolymers'. *J. Chem. Phys.*, 108(10) 4333–4339 (1998).
doi:10.1063/1.475834

[219] D. Chetverikov and A. Hanbury. 'Finding defects in texture using regularity and local orientation'. *Pattern Recognition*, 35 203–218 (2002).
URL http://aramis.ipan.sztaki.hu/strucdef/strucdef.html

[220] D. Chetverikov. 'Pattern regularity as a visual key'. *Image and Vision Computing*, 18 975–985 (2000)

[221] C. Upson and M. Keeler. 'V-buffer: visible volume rendering'. *ACM SIGGRAPH Computer Graphics*, 22(4) (1988).
doi:10.1145/378456.378482

[222] O. Wilson, A. van Gelder, and J. Wilhelms. 'Direct volume rendering via 3D textures'. Technical Report UCSC-CRL-94-19, University of California, Santa Cruz (1994).
URL http://www.cse.ucsc.edu/research/reports/ucsc-crl-94-19.ps.Z

[223] T. McReynolds and D. Blythe. 'Advanced graphics programming techniques using OpenGL'. SIGGRAPH '98 Course.
URL http://www.opengl.org/resources/tutorials/advanced/advanced98/notes/

[224] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit: An Object Oriented Approach to 3D Graphics*. Kitware, Inc., 3rd edition (2003).
ISBN 1930934076.
URL http://www.kitware.com

[225] J. Kniss, G. Kindlmann, and C. Hansen. 'Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets'. In 'Proceedings of the conference on Visualization '01. San Diego, California', pages 255–262. IEEE Computer Society (2001).
URL http://www.cs.utah.edu/%7ejmk/papers/vis01/

[226] J. Kniss, G. Kindlmann, and C. Hansen. 'Simian'.
URL http://www.cs.utah.edu/ %7ejmk/ simian/

[227] P. J. Love, P. V. Coveney, and B. M. Boghosian. 'Three-dimensional hydrodynamic lattice-gas simulations of domain growth and self-assembly in binary immiscible and ternary amphiphilic fluids'. *Phys. Rev. E*, 64(021503) (2001). doi:10.1103/PhysRevE.64.021503

[228] B. M. Boghosian, J. Yepez, P. V. Coveney, and A. Wagner. 'Entropic lattice Boltzmann methods'. *Proc. R. Soc. Lond. A*, 457 717 (2001).
URL http://arxiv.org/abs/cond-mat/0005260

[229] B. M. Boghosian, P. J. Love, P. V. Coveney, I. V. Karlin, S. Succi, and J. Yepez. 'Galilean-invariant lattice-Boltzmann models with H-theorem'. *Phys. Rev. E*, 68(025103(R)) (2003).
doi:10.1103/PhysRevE.68.025103

[230] I. V. Karlin and A. N. Gorban. 'Maximum entropy principle for lattice kinetic equations'. *Phys. Rev. Lett.*, 81(1) 6–9 (1998)

[231] S. Succi. *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford University Press (2001). ISBN 0-19-850398-9

[232] E. S. Boek, J. Chin, and P. V. Coveney. 'Lattice Boltzmann simulation of the flow of non-Newtonian fluids in porous media'. *Int. J. Mod. Phys. B*, 17(1–2) 99–102 (2003).
doi:10.1142/S021797920301714X

[233] M. Nekovee, J. Chin, and P. V. Coveney. 'Massively parallel mesoscopic simulations of complex fluids'. In 'Proceedings of the Simulation 99 UCLse/UK Conference on Simulation', (1999)

[234] M. Nekovee, J. Chin, N. González-Segredo, and P. V. Coveney. 'A parallel lattice-Boltzmann method for large scale simulations of complex fluids'. In 'Proceedings of the Fourth UNAM Supercomputing Conference, Singapore'. (1999).
URL http://www.chem.ucl.ac.uk/ccs/unam.ps.gz

[235] J. Harting, M. Venturoli, and P. V. Coveney. 'Large-scale grid-enabled lattice-Boltzmann simulations of complex fluid flow in porous media and under shear'. *Phil. Trans. R. Soc. Lond. A*, in press (2004)

[236] C. Pan, J. F. Prins, and C. T. Miller. 'A high-performance lattice Boltzmann implementation to model flow in porous media'. *Comp. Phys. Comm.*, 158(2) 89–105 (2004).
doi:10.1016/j.cpc.2003.12.003

[237] S. Ansumali, I. V. Karlin, and H. C. Öttinger. 'Minimal entropic kinetic models for hydrodynamics'. *Europhys. Lett.*, 63(6) 798–804 (2003). doi:10.1209/epl/i2003-00496-6

[238] N. S. Martys and J. G. Hagedorn. 'Multiscale modeling of fluid transport in heterogeous materials using discrete Boltzmann methods'. *Materials and Structures*, 35(254) 650–659 (2002)

[239] S. Donath. *On Optimized Implementations of the Lattice Boltzmann Method on Contemporary High Performance Architectures.* Bachelor thesis, Friedrich Alexander-Universität Erlangen-Nürnberg (2004).
URL `http://www10.informatik.uni-erlangen.de/Publications/Theses/Donath.p`

[240] R. Srinivasan. 'XDR: External data representation standard'. Network working group request for comments: 1832, Internet Engineering Task Force (1995).
URL `http://www.ietf.org/rfc/rfc1832.txt`

[241] R. Srinivasan. 'RPC: Remote procedure call protocol specification version 2'. Network working group request for comments: 1831, Internet Engineering Task Force (1995).
URL `http://www.ietf.org/rfc/rfc1831.txt`

[242] B. Callaghan, B. Pawlowski, and P. Staubach. 'NFS version 3 protocol specification'. Network working group request for comments: 1813, Internet Engineering Task Force (1995).
URL `http://www.ietf.org/rfc/rfc1813.txt`

[243] B. W. Kernighan and D. M. Ritchie. *The C Programming Language.* Prentice-Hall, second edition (1988). ISBN 0-13-110362-8.
URL `http://cm.bell-labs.com/cm/cs/cbook/`

[244] Message Passing Interface Forum. 'MPI: A message-passing interface standard'.
URL `http://www.mpi-forum.org/`

[245] 'The FLXmitter SPU'.
URL `http://www-unix.mcs.anl.gov/ %7ejones/ Chromium/ flxspu.htm`

[246] 'GNU autoconf'.
URL `http://www.gnu.org/software/autoconf/`

[247] R. E. Kaufman. *A Fortran coloring book.* MIT Press (1978). ISBN 0-262-61026-4

[248] The Open Group . 'Single UNIX specification, version 2' (1997).
URL `http://www.unix.org/version2/online.html`

[249] B. W. Kernighan. 'Why Pascal is not my favorite programming language'. Computing science technical report no. 100, AT&T Bell Laboratories, Murray Hill, New Jersey 07974 (1981).
URL `http://cm.bell-labs.com/cm/cs/cstr/100.ps.gz`

[250] F. van Hoesel. 'Xdrf'.
URL http://rugmd4.chem.rug.nl/hoesel/xdrfman.html

[251] National Center for Supercomputing Applications. 'HDF5 home page'.
URL http://hdf.ncsa.uiuc.edu/HDF5/

[252] W. Cunningham. 'The Portland Pattern Repository'.
URL http://c2.com/cgi/wiki?WelcomeVisitors

[253] K. Roy. 'LB3D on SGI Origin 3000 - evaluation and feedback report'. Internal report, CSAR (2003)

[254] E. Breitmoser. 'RealityGrid WP 2: Terascale regime scalability: Task T2.3: LB3D, benchmarking for the gold star rating'. RealityGrid technical report, EPCC (2003)

[255] M. Frigo and S. G. Johnson. 'FFTW: The fastest Fourier transform in the west'.
URL http://www.fftw.org/

[256] 'The Comprehensive Perl Archive Network'.
URL http://www.cpan.org/

[257] T. Jenness and S. Cozens. *Extending and Embedding Perl.* Manning Publications (2002)

[258] W. Gu, J. Vetter, and K. Schwan. 'An annotated bibliography of interactive program steering'. *ACM SIG-PLAN Notices*, 29(9) 140–148 (1994).
URL http://doi.acm.org/10.1145/185009.185038

[259] 'The RealityGrid project'.
URL http://www.RealityGrid.org

[260] J. M. Brooke, P. V. Coveney, J. Harting, S. Jha, S. M. Pickles, R. L. Pinning, and A. R. Porter. 'Computational steering in realitygrid'. In 'Proceedings of the UK e-Science All Hands Meeting, September 2–4', (2003).
URL http://www.nesc.ac.uk/events/ahm2003/AHMCD/pdf/179.pdf

[261] N. Margolus, T. Toffoli, and G. Vichniac. 'Cellular-automata supercomputers for fluid-dynamics modeling'. *Phys. Rev. E*, 56(16) 1694–1696 (1986).
doi:10.1103/PhysRevLett.56.1694

[262] N. Margolus. 'CAM-8: a computer architecture based on cellular automata'.
URL http://arxiv.org/abs/comp-gas/9509001

[263] M. Woo, J. Neider, and T. Davis. *OpenGL Programming Guide.* Addison-Wesley, second edition (1997). ISBN 0-201-46138-2.
URL http://rush3d.com/reference/opengl-redbook-1.1/

[264] Z. Fan, F. Qiu, A. Kaufman, and S. Yoakum-Stover. 'GPU cluster for high performance computing'.
URL http://www.cs.sunysb.edu/%7evislab/projects/urbansecurity/GPUcluster

[265] W. Li, X. Wei, and A. Kaufman. 'Implementing lattice Boltzmann computation on graphics hardware'. *The Visual Computer*, 19(7–8) 444–456 (2003).
URL http://www.cs.sunysb.edu/%7evislab/projects/amorphous/WeiWeb/hardwa

[266] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, CA (1999). ISBN 1-55860-475-8

[267] F. Berman, G. C. Fox, and A. J. G. Hey, editors. *Grid Computing: Making the Global Infrastructure a Reality*. J. Wiley and Sons (2003). ISBN 0-470-85319-0.
URL http://www.grid2002.org/

[268] I. Foster. 'What is the Grid? A three point checklist' (2002).
URL http://www-fp.mcs.anl.gov/ foster/Articles/WhatIsTheGrid.pdf

[269] E. Cerami. *Web Services Essentials*. O'Reilly and Associates (2002). ISBN 0-596-00224-6

[270] I. Foster and C. Kesselman. 'Globus: A toolkit-based grid architecture'. In I. Foster and C. Kesselman, editors, 'The Grid: Blueprint for a New Computing Infrastructure', page 259. Morgan Kaufmann (1999)

[271] J. Chin and P. V. Coveney. 'Towards tractable toolkits for the grid: a plea for lightweight, usable middleware'. UK eScience Technical Report UKeS-2004-01, UK National e-Science Centre (2004).
URL http://www.realitygrid.org/lgpaper.html

[272] W. S. Means and E. R. Harold. *XML in a Nutshell*. O'Reilly and Associates (2001). ISBN 0-596-00058-8

[273] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, D. Snelling, and P. Vanderbilt. *Open Grid Services Infrastructure (OGSI) (draft)*. OGSI Working Group of the Global Grid Forum (2003).
URL http://www.gridforum.org/Meetings/ggf7/drafts/draft-ggf-ogsi-gridservice-23_2003-02-17.pdf

[274] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe. 'The WS-Resource framework' (2004).
URL http://www.globus.org/wsrf/specs/ws-wsrf.pdf

[275] Manchester Computing Supercomputing, Visualization, and e-Science Group. 'The RealityGrid computational steering system'.
URL http://www.sve.man.ac.uk/Research/AtoZ/RealityGrid/Steering

[276] 'The Chromium project'.
URL http://www.sourceforge.net/projects/chromium

[277] 'OpenGL Vizserver 3.0: Application-transparent remote interactive visualization and collaboration'.
URL http://www.sgi.com/software/vizserver/

[278] 'Access Grid'.
URL http://www.accessgrid.org/

[279] P. V. Coveney and M. J. Harvey. 'Europe: AG aids RealityGrid and the TeraGyroid project'. *AG Focus*, 2(1) 4 (2004).
URL http://www.chem.ucl.ac.uk/ccs/AG_Focus_Spring_04_Final.pdf

[280] J. Oikarinen and D. Reed. 'Internet Relay Chat Protocol'. Network working group request for comments: 1459, Internet Engineering Task Force (1993).
URL http://www.ietf.org/rfc/rfc1459.txt

[281] P. Gutmann. 'PKI: It's not dead, just resting'. *IEEE Computer*. 25(8) 41–49 (2002)

[282] B. Beckles. 'Removing digital certificates from the end-user's experience of grid environments'. In 'Proceedings of the UK e-Science All Hands Meeting'. (2004).
URL http://www.allhands.org.uk/2004/proceedings/papers/250.pdf

[283] P. Gutmann. 'Plug-and-play PKI: A PKI your mother can use'. In 'Proceedings of the 12th USENIX Security Symposium', pages 45–58 (2003).
URL http://www.usenix.org/events/sec03/tech/gutmann.html

[284] M. McKeown. 'OGSI::Lite - an OGSI implementation in Perl' (2003).
URL http://www.sve.man.ac.uk/Research/AtoZ/ILCT

[285] P. Coveney, J. Vicary, J. Chin, and M. Harvey. 'Introducing WEDS: a WSRF-based environment for distributed simulation'. UK eScience Technical Report UKeS-2004-07, UK National e-Science Centre (2004).
URL http://www.nesc.ac.uk/technical_papers/UKeS-2004-07.pdf

[286] 'The Open Middleware Infrastructure Institute'.
URL http://www.omii.ac.uk/

[287] R. P. Gabriel. 'The rise of "worse is better"'.
URL http://www.jwz.org/doc/worse-is-better.html